Chapter 7
The Role of Multisensor Environmental Perception for Automated Driving

Robin Schubert and Marcus Obst

7.1 Introduction

Automated driving is characterized by a computer-based derivation and execution of appropriate driving maneuvers based on the current traffic situation. The basis for evaluating the traffic situation, however, is knowledge about all relevant entities in a vehicle’s environment, including traffic participants (vehicles, pedestrians), road infrastructure (lane markings, traffic signs, traffic lights), or obstacles (curbstones, potholes, bushes).

For acquiring such knowledge, numerous sensors are being used that permanently provide relevant data. These data, however, are typically not directly used by the high-level driving functions (such as path planning or maneuver decision-making). Instead, an intermediate processing layer is used that is referred to as the environmental model (see Fig. 7.1). This additional layer is used for several reasons:

- **Sensor errors**: Data directly provided by the sensors are subject to different errors. On the one hand, the measured signals can be disturbed by effects such as sensor noise, latencies, or calibration errors. On the other hand (and more severe), sensors may fail to detect objects at all or falsely report objects that are not present in the environment. Those effects are called false-negative or false-positive detections, respectively.

  The environmental model contains algorithms to reduce those errors by temporal filtering and to estimate the magnitude of errors that need to be considered when using its result for taking driving decision. Thus, environment modeling helps to increase the reliability of the perception.

R. Schubert (✉) • M. Obst
BASELABS GmbH, Chemnitz, Germany
e-mail: robin.schubert@baselabs.de; marcus.obst@baselabs.de

© Springer International Publishing Switzerland 2017
D. Watzenig, M. Horn (eds.), Automated Driving,
DOI 10.1007/978-3-319-31895-0_7
Fig. 7.1 The perception layer constitutes an interface between sensors and high-level functions

- **Sensor abstraction**: The sensor configuration of different vehicles will most likely not be the same. In addition, interfaces and properties of the utilized sensors may change. By providing a separating layer between the sensors and the driving function, the environmental model facilitates the implementation of driving functions that are independent from particular sensors. This has a huge impact on reusability and architectural quality of those functions. The environmental model may, thus, be considered as a *sensor-abstraction layer*.\(^1\)

- **Data fusion**: For many driving functions, information from different sensors are required (e.g., the velocity of an object measured by a radar and its class determined by a camera). The environmental model inherently combines data from all available sensors in order to provide both a higher quantity and a higher quality of available information. While the former is achieved by combining heterogeneous sensors or sensors covering different fields of view, the latter effect is based on sensor redundancy. The environmental model ensures that the driving functions can use a consolidated representation of the environment.

- **Modeling and hypothesis generation**: The required reliability of information representing the environment of the vehicle constitutes an inherent conflict when compared to the quality of the available sensor data. The decisive strategy for bridging this gap is to use *model knowledge*. In fact, the concept of the

\(^1\)To some extent, this can be compared to hardware abstraction layers used in programming to support modular software architectures that facilitate reusability.
environmental model is to form expectations of the environment (often in form of numerous hypotheses) and to confirm or falsify these hypotheses by calculating their fit to the observed sensor data. For instance, lanes may be assumed to be parallel and to follow a clothoidal shape. It is many grades of magnitudes easier to estimate the parameters of a clothoid based on noisy data than to determine a completely arbitrary shape from these data. The same concept holds for other entities such as vehicles, whose stable detection and tracking is only possible because it is assumed that they comply with typical motion patterns. Modeling also facilitates the estimation of quantities that are not directly observable. In fact, this concept forms the basis of the term environmental model, as it emphasizes that the main role in this layer is to define the expected structural representation of the environment and to estimate the parameters of this representation based on sensor data.

• **Prediction**: For automated driving, it is not sufficient to describe the current traffic situation. In fact, maneuver decisions or warnings to a driver should be issued in advance of a collision or another dangerous situation occurs. This implies the need to predict the traffic situation to the future in order to anticipate the consequences of driving decisions. The environmental model provides this functionality by exploiting model knowledge as described before. For instance, if typical motion patterns of vehicles are modeled and their dynamic parameters (such as their velocities and yaw rates) are estimated, future positions can be determined. As predictions can never be certain, however, also the reliability of such predictions needs to be quantified by the environmental model.

• **Integrity**: Together with the representation of the environment, probably the most relevant role of the environmental model is to provide quantitative indicators that describe its uncertainties. For a single sensor, it is difficult to determine the current error of its measurement. The environmental model, however, can exploit the redundancy between sensors as well as the model knowledge to estimate the current estimation errors. Such an error indicator is considered to fulfill the property of integrity if the true error never exceeds the estimated one. As automated driving functions are safety critical, this property is crucial for a safe operation. Only if the driving functions know about the current expected error of the environmental representation, it can take appropriate decisions. The most common way to describe these uncertainties is by means of probabilistic quantities. For this reason, probabilistic inference algorithms are at the heart of environmental modeling algorithms.

### 7.2 State of Practice

In general, the vehicle environment for automated driving applications comprises dynamic and static entities. In the following sections, typical methods to model and estimate these entities are presented.
7.2.1 Dynamic Environments

Dynamic environments are characterized by numerous entities that can move independently from each other. As described in Sect. 1, it is advantageous to model this motion mathematically in order to improve the robustness of the perception. For this reason, the most common approach for modeling dynamic environments is to apply an object-based paradigm by handling each moving object independently.

A commonly applied technique to that end is multiple objects tracking (MOT). Here, object tracking refers to the task of utilizing noisy measurements from heterogeneous perception sensors to derive both the number and the characteristics of relevant objects by filtering the sensor data over time. The characteristics are subsumed in a vector called the state and typically include continuous parameters such as position, velocity, and heading angle as well as discrete states like the object class.

Though there is a variety of MOT algorithms, many of them can be generalized to the structure illustrated in Fig. 7.2. The basic assumption of an MOT algorithm is that, due to false positives and false negatives, detections reported by sensors do not necessarily correspond to true objects. This inherent property of perception systems is taken into account by introducing the concept of tracks representing object hypotheses. A track consists of a state estimate (typically including a position and additional kinematic parameters) and a probability which quantifies the belief in the hypothesis that this track represents a true object (referred to as the probability of existence). Both the state estimate and the probability of existence are iteratively updated whenever new sensor data arrive.

The state of a track is modeled independently from the sensor measurements, while the relation between both is modeled by a mathematical function called the sensor model. By using several sensor models, the tracks can be updated by different sensors independent from each other. Thus, all sensors that are related to the state by a sensor model contribute to the estimation of this state as well as the estimation of the track’s probability of existence. This concept implicitly facilitates the spatial
and temporal alignment of data from different sensors—commonly referred to as (multiple sensor) data fusion.

The concept of MOT will be explained in the following subsections based on the structure illustrated in Fig. 7.2.

7.2.1.1 State Estimation by Bayesian Filtering

In general, several approaches to realize multisensor fusion are available [1]. A well-known and successfully applied technique in online multisensor fusion applications is Bayesian state estimation. By that approach, the temporal correlation between consecutive observations from multiple sensors can be continuously exploited and combined with a priori model knowledge. Finally, the result is represented by a probability density function (PDF). The recursive Bayes filter algorithm comprises two main parts:

- **State prediction**: In this step, the knowledge of the system under observation from the last time step $t_{k-1}$ is synchronized to the measurement time $t_k$ of the most recent sensor observations. This is achieved by applying a system model (e.g., an appropriate motion model [2] in the case of a vehicle state estimation problem) which describes the expected evolvement of the system over time. As this state synchronization is solely based on model knowledge (which typically does not match reality in every detail), it typically adds uncertainty to the estimate. Due to the fact that the state of the system under observation and the provided sensor measurement are often not defined in the same coordinate system, the predication step also includes an additional transformation process from the synchronized system state to the measurement domain. In order to achieve this transformation, a sensor-specific measurement model which considers sensor noise is applied.

- **State update**: After the prediction step, a probabilistic representation of the synchronized system state (also called a state hypothesis) described in the measurement domain and the particular sensor observation sample is available for comparison. Probabilistically speaking, the likelihood of the sensor model given the assumption of the state hypothesis is evaluated at the position of the observed measurement. Based on the result, the predicated state estimate is adjusted and considered the best knowledge of the system at time $t_k$. The result of this step is a probabilistic combination of the information from the latest sensor measurement and the previously accumulated system state.

Recursive Bayes filtering is an efficient and consistent technique to perform a probabilistic data fusion among multiple sensors for real-time applications. Different measurement principles and the sensor noise are considered by particular measurement models, while the alignment of asynchronous sensor data is implicitly achieved by applying the system model. It is worth mentioning that there are multiple practically relevant implementations of recursive Bayes filters: For linear systems whose uncertainties can be modeled by Gaussian PDFs, the Kalman filter is the optimal estimator. The practically most relevant filters are the extended
Kalman filter and the unscented Kalman filter for nonlinear systems that are subject to unimodal disturbances which can be approximated by Gaussian PDFs. For multimodal PDFs or in case of severe nonlinearities, particle filters can be applied [3].

7.2.1.2 Data Association

In order to update a track’s state estimate, sensor measurements are used. In practice, however, this process is subject to severe ambiguities as illustrated in Fig. 7.3. The output of a sensor is a list of detections within the field of view that are not related in any way to the object hypotheses that are handled by the MOT. For updating the state of a track, it is crucial to know which of these detections shall be used for updating a particular track. One part of the ambiguity related to this issue stems from the uncertainty of the state estimate itself. As the track’s position is not exactly known, it is difficult to distinguish between a detection that displays a different position than the track due to measurement noise and a detection coming from an adjacent object. The other part of the ambiguity is caused by the fact that sensor detections can be erroneous. In Fig. 7.3, some of the sensor detections may be related to the vehicle, they may be false positives, or they may represent a vehicle which is not yet tracked. Solving this ambiguity is referred to as data association (or, more precisely, measurement to track association) and can be considered the key challenge in multiple objects tracking.

The basis of data association is some kind of distance metric between the tracks and the measurements. Instead of using a geometrical measure such as the Euclidean distance, MOT requires to apply a statistical distance measure that, on the one hand, accounts for the uncertainties of the tracks and the sensor measurements and, on the other hand, facilitates the combination of different state variables such as
The role of multisensor environmental perception for automated driving involves combining position and velocity into a single metric. The metric of choice for this application is the **Mahalanobis distance**, which is a statistical measure describing the distance between a multi-dimensional probability density function and a point [4]. If the track’s state is represented by a Gaussian PDF, the metric can be considered as a track’s likelihood evaluated at the position of the measurement.

Based on the Mahalanobis distance between each track and each measurement, the measurement to track association takes place in several steps.

The first step is called **gating**. In this phase, detections that exceed a predefined distance to the closest track are excluded from the further steps. Those detections are considered as **non-associable**, which means it is considered very unlikely that those detections represent objects which are already being tracked. Instead, those detections are assumed to be potential objects that are not yet tracked—hence, they are used to create new object hypotheses (i.e., new tracks). In addition to using those non-associable detections as a source for new tracks, another rationale for gating in combination with decision-based association techniques is to avoid unreasonable association decisions.

Regarding the second step after the gating, there are two groups of techniques: The first one reduces complexity by conducting a unique n-1-pre-association in the sense that every detection is associated unambiguously to the closest track (which implies that several detection can be pre-associated to each track). Those approaches can be considered as **local** association techniques. The second group does not make use of this simplification, which makes it more performant in ambiguous situations but also more computationally complex.

In the final association step, again two different categories may be distinguished:

- **Decision-based data association**: In the first category, a unique association decision is derived in the sense that not more than one detection is associated to a track and vice versa. This can be achieved by minimizing the distance between tracks and detections. The local algorithm in this category is the **local nearest neighbor** association which determines the closest detection for each track. The **global nearest neighbor** association, however, enumerates overall association hypotheses and chooses the one with the smallest sum of distances.

- **Probabilistic data association (PDA)**: In this category, no hard decisions are made. Instead, all detections (possibly pre-associated to a track) are considered valid association hypotheses, and a probability is derived for each hypotheses which is higher for detections closer to certain tracks than for detections that are farther away. Depending on whether a pre-association has been applied, either a local variant (PDA) or a global variant (joint PDA—JPDA) of this approach can be used.

PDA and JPDA can be further distinguished into several subclasses of association algorithms depending on certain assumptions and models:

- **PDA/JPDA vs. IPDA/JIPDA**: The basic PDA/JPDA is using the association weights for the state update but not for updating the existence probability. An
extension is the integrated PDA (or the corresponding joint integrated PDA) that updates also the existence probability accordingly [5].

- **GPDA**: The basic PDA assumes that each object cannot generate more than one detection. This assumption limits the number of hypotheses and, thus, reduces complexity. However, it also implies that if several measurements are received for one object, all but one are per definition considered false positives. In some scenarios (like tracking of large trucks with a radar sensor), this assumption does not hold. The generalized PDA (that is also available in the integrated and joint variant) relaxes this assumption at the costs of a higher computational complexity [6].

As an interim conclusion, data association is a complex topic that can be solved by numerous algorithms that are summarized in the Tables 7.1 and 7.2. The choice of the right algorithm depends on the sensors as well as on the scenarios and the available computation power.

### Table 7.1 Systematization of data association techniques

| Gating | Data association                                                                 |
|--------|----------------------------------------------------------------------------------|
| Pre-association (per detection)    | n-1-association (“local”): Detections are associated to not more than one track |
| Deterministic vs. probabilistic association (per track) | n-m-association (“global”): Detections are associable to any track (no pre-association) |
| Deterministic vs. probabilistic association (per track) | **Local nearest neighbor** (hard 1-1 decision) |
| Deterministic vs. probabilistic association (per track) | **Probabilistic data association (PDA)** (no decision) |
| Deterministic vs. probabilistic association (per track) | **Global nearest neighbor** (hard 1-1 decision) |
| Deterministic vs. probabilistic association (per track) | **Joint probabilistic data association (JPDA)** (no decision) |

\[a\]This approach is called *data-driven* track creation. While it is the practically most relevant approach, the theoretically consistent way of creating tracks is to draw them randomly from a birth distribution

### Table 7.2 Further systematization of probabilistic data association techniques

| (Joint) probabilistic data association | Impact on existence estimation | Assumed number of possible true detections from an object |
|---------------------------------------|--------------------------------|----------------------------------------------------------|
| **Classical PDA/JPDA** | No influence between existence and association probabilities | Assumption of classical PDA/JPDA/IPDA/JIPDA: The sensor never receives more than one true detection from an object |
| **Integrated PDA/JPDA (IPDA/JIPDA)** | Mutual influence between both probabilities | **Generalized PDA/JPDA/IPDA/JIPDA**: A sensor can receive several true detections per object |
7.2.1.3 Existence Estimation

For the estimation of a track’s probability of existence, Bayesian filtering can be applied as well. However, in contrast to the estimation of a continuous state, the existence of a track is a binary variable. This simplifies the estimation, as it allows the usage of a discrete Bayes filter.

The belief that the track under consideration represents a true object is represented by the conditional probability $P(\exists x_k | Z_k)$ where the binary variable $\exists x_k$ denotes the existence of an object $x$ at time $k$ and $Z_k$ denotes the set of all measurements from time 0 to $k$. This conditional probability can be calculated using Bayes rule (using $\eta$ as a normalizing constant), which leads to

$$P(\exists x_k | Z_k) = P(\exists x_k | z_k, Z_{k-1})$$

$$= \eta \cdot P(z_k | \exists x_k, Z_{k-1}) \cdot P(\exists x_k | Z_{k-1})$$

$$= \eta \cdot P(z_k | \exists x_k) \cdot P(\exists x_k | Z_{k-1})$$

This equation contains two relevant terms: $P(\exists x_k | Z_{k-1})$ denotes the probability of existence conditioned on all measurements except the last one. It can be obtained by predicting the probability of existence from the last time step to the current time using the Chapman-Kolmogorov equation:

$$P(\exists x_k | Z_{k-1}) = \sum_{\exists x_{k-1}} P(\exists x_k | \exists x_{k-1}) \cdot P(\exists x_{k-1} | Z_{k-1})$$

The term $P(\exists x_{k-1} | Z_{k-1})$ corresponds to the probability of existence from the previous time step, which illustrates that this value can be calculated iteratively from each time step to the next. The conditional probability $P(\exists x_k | \exists x_{k-1})$ describes how likely it is that an existing object will persist or a new object will appear and is referred to as birth/persistence model. The far more relevant term, however, is the existence likelihood $P(z_k | \exists x_k)$. This term describes the expected sensor measurements for both for the case of an existing as well as a nonexisting object. Its usage depends on the selected data association strategy.

For a nearest neighbor association, the only relevant information is if a measurement could be associated to a track or not, that is, a binary variable. This reduces the existence likelihood to a probability distribution consisting of four values (of which only two are required due to the fact that probabilities sum up to unity). The required likelihoods are:

- $P(z_k = 1 | \exists x_k = 1)$: This is the true positive rate of the sensor, that is, the probability that the sensor will detect an existing object.
• $P(z_k = 1 | \exists x_k = 0)$: This is the false-positive rate of the sensor, that is, the probability that the sensor will provide detections from locations without existing objects.

Both quantities can be obtained by characterizing the sensor. If the true positive rate is considered zero outside the field of view, the fusion of several sensors with partially overlapping observation areas can be implicitly handled by updating the existence probability independently for each sensor. Both the true and the false-positive rate may vary over the field of view (for instance, the detection performance of a radar may depend on the azimuth angle due to its antenna characteristics).

In a system using PDA, the existence likelihood also depends on the true and false-positive rate of the sensors, but also on the distance between the measurement and the track. Further details can be found in [6].

7.2.1.4 Track Management

In MOT, a dedicated, application-specific strategy to inject and discard track hypotheses is required. A typical choice for the track initialization step is to apply a data-driven regime where new track hypotheses are proposed based on not associated sensor measurements after each sensor’s duty cycle. However, this approach assumes that the information from the observed sensor data is sufficient to perform the full inverse transformation from the measurement domain to the targeted system representation. If this requirement is not fulfilled (e.g., it is generally not possible to derive information about the absolute velocity of a vehicle from a radar measurement which only includes angle, range, and radial velocity) further assumptions have to be made.

The same applies to the discarding strategy of already confirmed object hypothesis. Again, it depends on the targeted application and particular use case which hypotheses are considered relevant and thus have to be kept and which can be safely removed (e.g., in order to save computational resources).

7.2.2 Static Environments by Occupancy Grid Mapping

Whereas the modeling of dynamic environments as introduced in Sect. 2.1 aims to represent individual entities, occupancy grid mapping is a complementary approach more suitable for arbitrary static environments which does not assume any type of feature. The concept of grid mapping is to model the environment as an evenly spaced two-dimensional grid\(^2\) with a fine-grained cell size. Each cell inside of the grid comprises a binary random variable that represents the presence of an

\(^2\)There are also other variants of grid implementations which consider more dimensions such as the 4D grid [8].
obstacle at that particular location. In order to estimate this occupancy from sensor measurements, the individual cells are continuously updated by applying a discrete Bayes filter algorithm similar to the existence estimation step in MOT of Sect. 2.1. Thus, the occupancy grid can be considered a storage for spatially and temporally distributed measurement data.

In contrast to dynamic environment modeling with MOT, which provides confirmed object hypotheses, the result of an occupancy grid can be efficiently used to infer information about free space areas. This is in particular important for decision-making and path planning of automated maneuvers.

Due to the lack of dynamic quantities in the estimate, occupancy grids are typically not well suited for dynamic environments. There are however, hybrid approaches such as particle grids that are attempting to overcome this limitation and to constitute an estimation approach for both static and dynamic environments [7].

7.3 Challenges in Data Fusion

In the previous section, typical techniques for modeling dynamic and static environments for automated vehicles have been presented. The following section introduces selected challenges which practitioners may encounter during the implementation of data fusion systems for environmental perception.

7.3.1 Sensor Characterization

Sensor observations are a crucial input of every perception system. However, in general, the sensors which are used to observe the environment have weaknesses. First, not every measurement necessarily belongs to a real existing object. Those false-positive measurements (whose number is time varying) have to be eliminated during the tracking process. Vice versa, not every existing entity generates a measurement at sensor cycle and thus leads to a false negatives. In addition, the measurements are disturbed by noise. Finally, sensors have a limited detection range and finite resolution.

For the successful implementation and application of a multisensor data fusion system such as MOT, a correct and realistic modeling of the previously mentioned sensor properties is inevitable. In the following text, common techniques to derive and specify these sensor properties are stated. Moreover, examples of sensor mismodeling and their influence to the environmental model will be given:

- **Measurement noise**: Random fluctuations in measured signals are referred to as measurement noise. They can be characterized by statistical parameters that describe the average deviation of a sensor observation from the true value over
time. For example, for a radar system that measures angle, range, and range rate, the measurement noise is usually given independently for each dimension and described by a zero-mean Gaussian distribution with appropriate variances. Finally, the realization of a sensor observation and the knowledge of its behavior is used by statistical filtering such as Bayesian estimation in order to reduce the uncertainty over time.

Usually, these parameters are either extracted from the sensor manual or derived via empirical testing under controlled conditions. Although initial values from a sensor manual are usually a good starting point, practice reveals that these parameters need to be tested and validated carefully within the target scenario and the utilized data fusion algorithm. In MOT, for example, the measurement noise directly influences the performance of the data association step and the accuracy of the estimated objects. If the measurement noise is set too small, the tracking algorithm will not be able to generate stable object hypotheses (i.e., object with constant track IDs). On the other hand, assuming a too conservative value for the sensor noise limits the performance of the environmental model to represent high-dynamic maneuvers.

- **Field of view**: As the concept of MOT is based on object hypotheses (see Sect. 2.1), the perception system naturally aims to predict whether and how a particular object hypotheses will be measured by a sensor system. Generally, speaking, a sensor observation is only expected inside the sensor’s field of view (FOV). Inside the FOV, a sensor observation might either correspond to a real object (and thus confirms its existence) or is considered clutter and should be ignored. If a sensor observation is missing for a predicted object hypothesis, this usually indicates that the hypothesis inside of the MOT is incorrect and should be removed. Of course, this decision is not made at one particular time step but derived over time. The time span necessary to make this decision directly depends on the detection behavior of a sensor.

A typical configuration in environmental perception for automated vehicles is to have multiple perception sensors realizing a partially overlapping FOV in order to increase robustness and ensure proper handover. It is worth mentioning that the consistent performance of MOT is rather sensitive to the correct modeling of the overlapping area. If sensor FOVs are modeled too optimistic (i.e., sensor range is modeled to large), the MOT algorithm will consider nonexistent sensor observations at the FOV border as potentially missed detection and thus propose to delete the object hypothesis. This may lead to the unintuitive situation where two sensors work against each other instead of confirming a common object hypothesis.

- **Detection behavior**: Although the measurement noise already gives an indication of the quality of a sensor observation in terms of accuracy, it does not include any information about the reliability of a sensor system. Similarly, while the sensor’s FOV already gives a binary indication whether a sensor observation can be expected at a particular position around the host vehicle or not, it does not quantify the probability of detection in a statistical way. A straightforward
The Role of Multisensor Environmental Perception for Automated Driving

approach is to quantify the probability whether a sensor will detect an object given its existence inside of the FOV with a constant detection probability.

Given an exemplary high-reliable sensor which usually always detects objects inside the FOV, already one missing detection might indicate that the object hypotheses inside the perception is wrong and should be removed.

Although the assumption of a constant detection probability yields reasonable results for simple setups and scenarios, practical evaluation have shown that most perception systems require a more sophisticated modeling of the detection probability. A typical evolution in MOT modeling is to make the detection probability dependent on the position inside of the sensor FOV. Again, empirical evaluations may be used to derive these values.

• **Occlusion:** In general, a particular sensor is able to detect an object inside its FOV with a given detection probability, may it be constant or dynamically derived from other parameters such as object position or weather conditions. However, for a certain class of perception sensors such as lidar, radar or imagery the FOV may be temporarily impaired due to occlusion of other objects. In order to consistently distinguish between the systematic outage caused by temporarily occlusions and missing detections according to the behavior described by the detection probability, the sensor model needs to explicitly consider the position and spatial extension of other track hypotheses. If an MOT implementation does not consider occlusion, objects which are going to be occluded will be typically instantly discarded from the track list and are consequently no longer part of the environmental model. Proper occlusion handling is especially important for advanced scenarios where communication-based information such as vehicle-to-vehicle data is considered inside of the environmental perception.

### 7.3.2 Extended Objects

As previously mentioned in Sect. 2.1, MOT aims to estimate the time-varying number and the characteristics of the dynamic road entities in the surveillance area in an integrated manner. Current practical MOT implementations already consider the fact that the sensor observations are in general not perfect and thus are subject to *noise* and *false positives*. Moreover, temporary outages—so called *false negatives*—are typically treated by modeling with an appropriate sensor-specific detection probability.

In addition, the core part of almost every MOT implementation, the data association algorithm, typically assumes that each object generates not more than one physical sensor detection. Basically this implies that the object to be tracked is considered a *point source* that has one unique and fixed reflection center. The rationale behind that choice is to ensure an efficient implementation with a computational complexity that can be handled under real-time conditions.

However, not in all situations, this strong assumption is appropriate. For example, Fig. 7.4 illustrates a scenario where a remote vehicle generates three true
observations on the rear bumper which is directly observable by the front-looking perception sensor of the host vehicle. Apparently, whether an object generates more than one true sensor observations directly depends on the sensor resolution and the dimensions of the object under observation. By taking into account that the performance of modern perception sensors constantly increases, the issue of multiple detections per object becomes more relevant for future applications. Moreover, automated systems are going to target denser environments such as urban areas where the average distance to other road users is lower, and thus the number of detections per object increases as well.

A straightforward approach in practical systems to address the challenge of multiple detections per object is to introduce a dedicated detector stage between the sensor interface and the perception layer. Typically a clustering approach is chosen which refines multiple observations of an extended object from one epoch to one particular object measurement. By that approach, the point source assumption is kept, and the usual data association algorithms can be applied. However, one major drawback of non-statistical methods such as clustering for object extraction is the still unsolved question of the consistent handling of measurement uncertainties and false-positive phenomena. In addition, the introduction of an explicit detection stage makes it nearly impossible to reverse that decisions once they are made using the shown approaches.

### 7.3.3 Track Initialization

The basic concept of MOT is to improve the reliability of the state estimate as well as the confidence on the existence of a track by iteratively updating it using measurements from several subsequent time steps. As described in Sect. 3, the main challenge for steady-state tracks is data association. However, another significant step that can have a crucial impact on the tracking performance is track initialization. Track initialization is required if new object hypotheses (tracks) are
created based on sensor data that is unlikely to be generated from existing tracks. Initialization of a track means to derive a first-state estimate from a single-sensor detection. If the quality of this initialization is insufficient, the track will most likely not be associated to measurements from subsequent time steps. This, in turn, will prevent a steady-state tracking and lead to a repeated creation and deletion of tracks representing the same physical object. In particular for applications where the time to confirm an object is particularly short (e.g., automatic emergency braking), such an effect can lead to severe functional failures.

The main challenge in track initialization is to find initial values for nonobservable quantities, mainly velocity and heading. The difficulty of this step substantially depends on the sensors and the scenarios in which the perception system shall be applied. This shall be discussed on two examples.

Cameras and lidars are capable of providing position measurements of relevant objects. They are not capable of directly measuring velocities. A radar, however, can measure velocities but only in radial direction. This means that the velocities of objects in front of the ego vehicle can be directly measured by the radar, while the longitudinal velocities of objects moving on one side of the ego vehicle cannot.

The heading angle of an object cannot directly be measured. The variability of this quantity, however, depends on the scenario. In a highway scenario, it can be appropriate to initialize objects to travel in the same direction as the host vehicle. In urban environments like intersections, however, this assumption is not valid. Instead, objects may move in arbitrary directions.

A common way to handle initialization ambiguities is to use multiple initialization hypotheses. In later time steps, those hypotheses can be evaluated using their fit to the additional sensor data available at that time. However, this approach requires a rigorous handling of existence probabilities in order to avoid violations of the system’s integrity in terms of correctly determining the confidence on the existence of an object.

### 7.3.4 Asynchronous Sensors and Out-of-Sequence Processing

One challenge in data fusion is the combination of independent measurements from multiple sensors to get a more accurate and reliable estimate of the environment. In general, sensors provide measurements in fixed time intervals. However, they are usually not synchronized with a common clock, and thus it is rather likely that the measurements are asynchronously arriving at the data fusion. Especially in dynamic environments, a synchronization strategy is needed as otherwise the true state of an object might have changed between the sensor observations. By using Bayesian filtering, as proposed in Sect. 2, measurement synchronization is naturally included by applying the prediction step of the system model.

---

3Save by high-resolution sensors such as lidars in close distances as a result of a clustering process.
Furthermore, the data fusion of several heterogeneous sensor systems raises the problem of correctly reordering the individual observations according to their time of validity. This problem commonly referred to out-of-sequence measurements (OOSMs) arises mainly due to different inherent sensor latencies [11]. In general, sensor latencies comprise acquisition time due to the measurement principle, processing time inside of the sensor, as well as the time required for communication toward the perception system. The latencies can also be classified to be either deterministic or nondeterministic. If these latencies are not properly handled inside of the multisensor fusion, the original order of the measurements is lost. Exemplarily sensor latencies of radar-based perception sensors are in the range of 80–200 ms [8].

However, a typical Bayes filter as used in MOT relies on the arrival of data in the correct chronological order and will therefore generate corrupted estimation results under an OOSM scenario without proper handling. Especially scenarios with high dynamics such as vehicle tracking (e.g., automatic emergency braking) are vulnerable to this problem.

Although there are several potential approaches with different levels of complexity to handle OOSMs, it is usually not obvious which technique should be implemented in a particular MOT application. A straightforward solution which gives reasonable results when the variance of the latencies among the perception sensors is rather low is to completely ignore the latency while slightly increasing the modeled measurement noise of the sensors. The rationale behind that approach is to transform the error in the time domain into the measurement domain. Other, more complex approaches, either aim to perform an extrapolation of the sensor measurements [9] or directly try to apply dedicated OOSM strategies such as retrodiction to include the measurements inside of the multisensor data fusion according to their correct time of validity [10].

7.4 Implementation Workflows and Paradigms for Perception

In the previous sections, the core algorithms as well as selected challenges of environmental perception systems have been presented. In this section, these contents shall be mapped on state-of-practice design paradigms and tool-based workflows that support the implementation process. This includes the implementation and configuration of perception software modules and their validation.

7.4.1 Design Paradigms for Perception Software

The straightforward approach for implementing a software component for environmental perception is to apply one of the commonly used models for system
design in the automotive area (such as the V-model) and to start the software implementation based on previously defined requirements. This implies using a software development environment that is suitable for embedded systems as the final software is supposed to be executed in an electronic control unit (ECU) of a vehicle.

While this approach is well suited for other parts of automotive software, it displays significant disadvantages for the implementation of environmental perception software. The reasons for this can be found in the challenges described in the previous section: Due to the variability of environmental perception with respect to sensors and scenarios, it is rather challenging to specify the utilized algorithms in advance. Starting the implementation immediately from the specification bears the risk of either failing to comply with those requirements or using an algorithmic configuration that is significantly more complex than required and, thus, wasting computational resources in the final system. Due to the lack of applicability of modern software design patterns to embedded software, also the reusability of such software for future systems is limited.

For these reasons, the most commonly used design paradigm is an iterative approach that is based on an early prototype of the system which is continuously tested and improved to refine the system requirements. The advantages of this approach are the higher development speed (due to the usage of a non-embedded environment) as well as the better extendibility and reusability of the implemented software. This paradigm, however, can be further divided: One common option is to implement the software prototype by manual coding. The sole advantage of this approach is the usage of a non-embedded programming language. A second approach is to use model-based paradigms building on predefined libraries and architectural models. This approach yields a significant acceleration of the development process, as it provides a wide range of pre-implemented algorithms and models. The transition to the series code can be done using automatic code generators, which makes the manual coding of embedded software obsolete. In addition to the advantages in development speed, automatic code generation is also widely accepted to provide safer code as typical manual coding errors can be avoided. The disadvantage is that (given sufficient efforts) manual embedded code typically can be more optimized for particular platforms and applications than auto-generated code.

Model-based design using auto-generated code is a well-established design paradigm for many automotive software components (for instance, engine ECU or controller design). The basis of this paradigm is the assumption that the software follows a domain-specific architecture and, thus, can be modeled in a unified way using a predefined set of building blocks. While this is true for many applications, the process of defining a standardized model for environmental perception is still ongoing. As of today, a major part of environmental perception algorithms can be modeled in a unified way; however, there are other parts that need to be adapted to the sensor configuration or the scenario, respectively.

In addition, classical model-based design tools are designed for signal-oriented systems and allow for handling and processing a set of scalar signals. The input of
perception systems, however, is typically more complex. Data types to be processed include object lists of variable length, point clouds, or images. The usage of signal-oriented tools for the processing of such data makes the design process particularly complex.

Based on these two limitations of current model-based design paradigms, an additional workflow is increasingly being used. This paradigm is referred to as hybrid prototyping in this chapter, as it combines characteristics of model-based and coding-based prototyping. The concept of hybrid prototyping is illustrated in Fig. 7.5: On the one hand, this paradigm assumes that the key algorithms described in Sect. 2 are provided by a perception library in the numerous available variants. By defining a generic architecture of environmental perception, such a library can be used to configure the algorithmic layer.

In a second layer (called the sensor and scenario layer), the variants are too numerous to be covered by one generic model. Thus, standardized interfaces can be defined that allow the user to implement important parts of the perception, such as the detection model and the track initialization, by adding his own code. Finally, the user may add additional code anywhere in the model. This may be necessary to handle special scenario conditions or to implement particular processing prioritization strategies.

The application of this paradigm requires a code generator that is not restricted to a limited set of models and templates, but facilitates a generic code generation from both library-provided models and user code. Table 7.3 summarizes the different design paradigms discussed in this section.

**Table 7.3** Comparison of design paradigms for perception software

| Design paradigm               | Transition to series system | Development time | Flexibility |
|-------------------------------|-----------------------------|------------------|-------------|
| Direct ECU implementation     | Not necessary               | Long             | None        |
| Coding-based prototyping      | Manual re-implementation    | Long             | Low         |
| Model-based prototyping       | Template-based code generation | Short          | Medium      |
| Hybrid prototyping            | Generic code generation     | Short            | High        |
7.4.2 Software Environments for Testing and Validation

Testing and validating software typically requires feeding defined input data, executing the software, and comparing the results to the expected output. For environmental perception systems, the dimensionality of the input data is significant. In addition, data for particular scenarios and use cases are required for testing and validation. Finally, the complexity of the system makes it particularly challenging to identify error sources without a domain-specific data representation and the calculation of dedicated key performance indicators (KPIs). These characteristics motivate the usage of domain-specific environments for implementing, debugging, and testing environmental perception software systems. In the following, key properties of such environments are discussed in order to provide guidelines for the reader to select an appropriate environment:

- **Sensor interfaces**: Testing perception software implies feeding it with data from automotive perception sensors such as radars, cameras, or lidars. Multisensor frameworks shall provide ready-to-use interfaces to typical sensors and automotive interfaces (including access to automotive bus systems). To facilitate the usage of proprietary or prototypical sensors, those sensor interfaces should be extendible by the user.

- **Data synchronization**: One key characteristic of perception systems is the usage of asynchronous sensors with different update rates. While some perception algorithms support an implicit data synchronization by predicting the state estimate to the time of the data to be processed, other techniques require a manual data synchronization. A testing environment should provide different configurable ways to synchronize data and active processing components.

- **Data recording/replay**: While tests and demonstrations in a vehicle in live (real-time) conditions are useful, debugging and manual validation requires different modes of feeding data into the perception software. State-of-practice multisensor environments facilitate the convenient recording of complex, timestamped data. This facilitates a synchronous replay including features such as pausing, single-step processing, or the selection of particular scenes.

- **Interfaces to simulation software**: Testing perception software requires sensor data. However, sensors and test vehicles are not always available at the beginning of a development process. Furthermore, not all scenarios can be safely tested in a real environment. For this reason, simulation software\(^4\) is available to allow for defining scenarios and run simulations. Most simulation tools facilitate closed-loop test in which the result of a sensor data processing is fed back to modify the behavior of the simulation. A testing environment for perception systems should provide interfaces to common simulation tools in order to facilitate virtual validation.

\(^4\)For instance, PreScan by TASS International or Carmaker by IPG
Visualization: Due to the complexity of perception systems, a domain-specific data representation is required to reasonably debug the system and identify errors. For manual debugging, a rich visualization is an invaluable tool for analyzing the perception performance. Typical requirements to that end include overlaying camera images with sensor data or perception results and visualizing the complete traffic scene in a top-down perspective. Figure 7.6 illustrates this notion on the example of a multisensor 360° perception.

Interfaces to other domain-specific software: Parts of software for automated vehicles are written in specific development environments, such as MATLAB Simulink for controller design. Testing tools for perception software should support interfaces to such environments in order to facilitate a unified testing of perception and controller software.

Test automation: In addition to manual debugging, testing perception systems typically requires the automated testing. This implies the automated processing of huge amounts of recorded sensor data as well as the calculation of specific KPIs. Testing environments should, thus, provide interfaces for external test management tools to facilitate selection of data to be processed, parameterization of components, and automated control of applications under test.

These requirements illustrate that the test and validation of environmental perception software requires domain-specific tool support to facilitate an efficient process. Experiences from practice indicate that the tools available for this purpose can contribute significantly to the implementation of reliable perception software.
7.5 Conclusions

The environmental model has been identified as a key component of automated vehicles in this chapter. Typical algorithms for static and dynamic environments, in particular multiple objects tracking, have been presented and discussed. In addition, additional information from a practitioner’s perspective have been given:

On the one hand, it has been argued that there is no generic perception for all sensor configurations and use cases, but rather a need for configurable perception software. This configuration needs to be based on the sensor configuration and the targeted use cases (which, in turn, influence the choice of appropriate algorithms). Thus, one conclusion of this chapter is that, for automated driving, the environmental model does not exist. Instead, we will most likely see a variety of implementations.

On the other hand, it has been shown that a configurable perception requires new design paradigms. Though classical manual coding or pure model-based design are possible implementation approaches, a much higher design efficiency can be achieved using state-of-the-art methodologies as presented in this chapter.

On the other hand, it has been shown that due to state-of-practice design paradigms and tool-based workflows, the efficiency of the design process for this software could be significantly increased over the last years.

References

1. B. Khaleghi et al., Multisensor data fusion: A review of the state-of-the-art. Inf. Fusion 14(1), 28–44 (2013)
2. R. Schubert et al., Empirical evaluation of vehicular models for ego motion estimation, in Intelligent Vehicles Symposium (IV), 2011 IEEE. IEEE (2011)
3. S. Thrun et al., Probabilistic Robotics (The MIT Press, Cambridge, 2005)
4. P.C. Mahalanobis, On the generalised distance in statistics. Proc. Natl. Inst. Sci. India 2(1), 49–55 (1936)
5. D. Musicki, R. Evans, Joint Integrated Probabilistic Data Association—JIPDA, in Information Fusion, 2002. Proceedings of the Fifth International Conference on, vol. 2, pp. 1120–1125, 8–11 Jul 2002
6. C. Adam, R. Schubert, G. Wanielik, Radar-based extended object tracking under clutter using generalized probabilistic data association, in Intelligent Transportation Systems—(ITSC), 2013 16th International IEEE Conference on, pp. 1408–1415, 6–9 Oct 2013. doi:10.1109/ITSC.2013.6661593
7. R. Danescu, F. Oniga, S. Nedevschi, Modeling and tracking the driving environment with a particle-based occupancy grid. Intell. Transp. Syst. IEEE Trans. 12(4), 1331–1342 (2011). doi:10.1109/ITSS.2011.2158097
8. C. Coué et al., Bayesian occupancy filtering for multitarget tracking: An automotive application. Int. J. Robot. Res. 25(1), 19–30 (2006)
9. M.M. Muntzinger et al., Reliable automotive pre-crash system with out-of-sequence measurement processing, in Intelligent Vehicles Symposium (IV), 2010 IEEE. IEEE (2010)
10. A. Rauch et al., Car2x-based perception in a high-level fusion architecture for cooperative perception systems. *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE (2012)

11. Y. Bar-Shalom, Update with out-of-sequence measurements in tracking: Exact solution. *Aerosp. Electron. Syst. IEEE Trans.* 38(3), 769–777 (2002)