1 Introduction

With the DARPA "Grand Challenges" [THR06] in 2004 and 2005 and the following DARPA "Urban Challenge" [URM07] in 2007, the feasibility for automated driving up to level 4 as defined by the Society of Automotive Engineers (SAE) [NN17] has been shown in controlled environments. This opened up the discussion how to revolutionize mobility.

The main challenge that can be addressed with automated vehicles is the reduction of the approximately 1.3 million people that die each year worldwide in traffic accidents according to WHO [ORG18]. Automated vehicles are less prone to human errors like distraction, fatigue or speeding. Additionally, the automated vehicle can provide faster reaction times with the increase of technology and a full 360° coverage of the surroundings given the correct sensor setup. Furthermore, automated vehicles offer the potential to reduce inner city parking demand [ZHA20a] and decrease the cost of ride hailing, enabling mobility e.g. for elder people and potentially reducing emissions by making personal cars obsolete [SEV21].

In order to realize automated driving, companies either chose the top-down or bottom-up approach. The top-down approach consists in over-equipping the vehicle with a multitude of sensors and compute power followed by a phase of iterative reduction and refinement to directly target SAE level 4-5 automation. Prominent examples of which are e.g. Waymo [SUN20] and ArgoAI [CHA19]. On the other hand, most "traditional" automakers like Ford, VW or BMW, often focused on iteratively improving their already existing automated functions (SAE level 1-2) to higher levels of automation while at the same time being open for collaborations with the top-down approaching companies [FOR20].

In order to automate more sophisticated driving tasks (e.g. parking or lane changes), the vehicle has to be capable to perform the basic steps of robotic systems, namely sense, plan and act [ARK98]. Based on the degree of planning, one can distinguish the reactive paradigm which uses fixed mappings between sense and act, hierarchical paradigm with online planning and hybrid paradigm with steps-wise online planning. An illustration of the hierarchical paradigm is depicted in Fig. 1-1. However, for all of those paradigms the sensing block plays a crucial role as it provides the baseline for all of the upstream tasks. This sensing block can be further divided into sensory processing followed by world modeling (see [ARK98]).

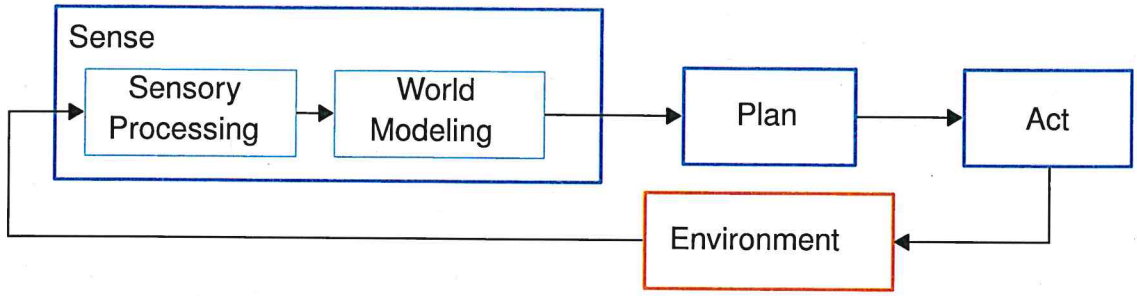

Fig. 1-1: Block diagram of a hierarchical robotic system.

The focus of this work lies on improving the world modeling capability of the sensing block for current and near future passenger vehicles. In order to perceive all obstacle locations in the full 360° surrounding of the ego vehicle, sensors like lidars, radars, cameras and ultrasonic sensors can be deployed (see e.g. NuScenes sensor setup [CAE20]). The obstacle locations, however, are only readily provided by spinning 360° lidars, which, nowadays, are too expensive to be deployed in passenger vehicles. An example of such a lidar is the Velodyne VLP16 (Puck) ranging currently around 4,000 $ according to [CNE18] and [VEL18b]. Therefore, these lidars are currently only deployed by companies focusing on automated driving (SAE level 4-5) or to provide ground truth information to verify automated driving features up to SAE level 3. For the remaining sensors, additional steps have to be performed like e.g. temporal accumulation to densify the measurements (mostly for radar and ultrasonic sensors) or domain adaption as applied to transform camera measurements into Bird's-Eye-View (BEV). It shall further be mentioned that, while being small, robust and cheap, ultrasonic sensors only provide information for narrow parking maneuvers since their range is restriction to about 6 m (e.g. [BOS22]). Therefore, they are still found in nowadays passenger vehicles but will not be included in this work.

Besides manually defining the transformation from sensor measurements to object locations, Neural Networks (NN) models are recently utilized to learn the transformation from data. This alleviates e.g. the problem of iteratively adapting the handcrafted model to every possible edge case while at the same time ensuring that the newly implemented changes do not interfere with the former performance of the model. Instead, a once trained NN model can be steadily improved by collecting edge cases, including them into the database and retraining the model. While this sounds tempting, there are also downsides to the application of NNs. Some of these downside include the lack of NNs to identify situations for which they have not been trained ("unknown unknowns"). In such situations, the predictions might not be based on semantically meaningful features and, as such, are sensitive to distributional shifts [APT19].

To mitigate these problems of NNs and verify them for production, the article "Safety

First for Automated Driving" [APT19] proposes the following. The NN can be extended to also predict its uncertainty. In this context, the aleatoric or data inherent and the epistemic or model related uncertainty have to be distinguished. The combination of these uncertainties can help downstream function modules to decide to which extent the predictions can be trusted. Additionally, the epistemic uncertainty can be used as an indicator whether to collect a data point for later retraining or not. Another way to verify NN predictions is the utilization of a so called observer. The observer checks the predictions of the NN e.g. given a set of rules like "Does the predicted trajectory coincide with any obstacle locations?". Eventually, a module can be run in parallel to the NN and predict the same target to provide redundancy. This can e.g. be realized using a slightly differently trained NN or one of the manually defined white box models.

In this work, the State-of-the-Art (SotA) of learned and handcrafted inverse sensor models is summarized in Chapter 2 and analyzed with regards to exemplary requirements found for nowadays passenger vehicle applications in Chapter 3. Based on these findings, Chapter 3 continues to detail extensions to potentially improve the security of the best suited SotA NN candidate following the guideline of [APT19]. More specifically, both SotA as well as newly proposed uncertainty estimation techniques are defined to modify the NN's predictions. Furthermore, a novel temporal fusion approach is defined in Chapter 3 to tackle the challenges that occur when temporally accumulating the learned environment models used in this work. At the end of Chapter 3, a novel alternative temporal fusion is defined in which the NN estimates have to be first verified by the handcrafted model. In this scenario, the handcrafted model acts as an observer for the NN predictions which should potentially further strengthen the security aspect according to [APT19].

To analyze the benefits and downsides, both learned and handcrafted benchmark models are tuned for the investigated task in Chapter 4. Afterwards, the benchmark models are compared against each other for each sensor modality and against the proposed uncertainty extensions. Similarly, Chapter 5 contains the experiments to compare the SotA temporal fusion with the proposed variants to highlight the need for the proposed novel fusion approach as well as show its limitations. Eventually, Chapter 6 concludes the work with a discussion of the obtained results with respect to the imposed requirements and an outlook for further work.

It is expected that these improvements have the most benefit in the passenger vehicle domain given that high accuracy, dense 360° lidar sensors cannot be deployed in this domain at least within the next decade. These findings are aimed to improve the perception block of the robotic pipeline (see Fig. 1-1) and, as such, will improve the performance of all downstream functions increasing the overall robustness of automated driving features.

In the normalized form, $m(\emptyset)$ is always zero and, thus, can be removed from consideration. The vector of normalized mass functions can then be written as follows

$$\boldsymbol{m} = \begin{bmatrix} m(F), & m(O), & m(U) \end{bmatrix}^\mathsf{T} = \begin{bmatrix} m_f, & m_o, & m_u \end{bmatrix}^\mathsf{T} \qquad \text{Eq. 2-7}$$

The evidential formulation adds and additional degree of freedom by modeling the unknown class separately. This allows to distinguish the case of conflicting information, indicating a cell to be free and occupied at the same time, from the case of absent information which, in the probabilistic view, both results in p_o equaling 0.5. In [MOR11, YU15, KUR12], it is proposed to utilize the conflicting information to represent dynamic objects which comes naturally since they are neither free nor occupied but rather in a transition state.

To build such occupancy maps, sensor measurements are being fed into so called ISMs to obtain an estimate of the occupancy state around the vehicle. This estimate is then transformed into map coordinates and fused into the map using evidential combination rules. To obtain unbiased maps in which each information is weighted equally, the ISM estimates have to be informational independent [PAG96]. The following sections will elaborate on the variants of ISMs and evidential combination rules at hand.

2.2 Geometric Inverse Sensor Models

In contrast to sensor models, which describe the sensor characteristics given the environment, ISMs describe the environment given the sensor measurements. These models can be divided into two categories. On the one hand, the physical measurement principles of the sensor can be used to define a geometrical relationship between the measurement and the environment. These models will be referred to as geo ISMs. On the other hand, data-driven methods can be applied to learn the ISM from large bodies of data. Even though it is possible to use other data-driven methods to learn ISMs, the literature mainly focuses on the application of deep artificial neural networks. These models, which will be referred to as deep ISMs, are specifically suited for the task at hand, since they excel over other data-driven methods when it comes to utilizing large amounts of data [ZHO14] and are universal function approximators [HOR91].

2.2.1 Ray-Casting in geo ISMs

Most of the sensors deployed in automated driving for environment perception use a radial sensing principle like e.g. cameras and lidars. These sensors are only capable of perceiving objects in direct line of sight. To describe these sensor models, the following notation shall be introduced. For the geo ISMs, a Gaussian measurement noise is assumed whose mean and variance are defined in polar coordinates $[e_r, e_\varphi]$ as (μ_r, μ_φ)

surface, is the application of a height threshold as e.g. proposed in [THR06]. This, however, often leads to artifacts since the ground in most environments has a non-zero curvature. Solutions for this problem, as described in [NAR18], range from fitting either a single plane or piece-wise planar model e.g. using RANSAC or the Hough transform [FIS81, HOU62, OLI16, TIA20] and classifying all points within a given distance as ground, performing the classification based on thresholds of local features like average height, average variance, deviation in region normal vectors etc. [LI14, ASV15], through to the application of Conditional Random Fields [RUM17], Markov Random Fields [GUO11], deep learning [ZHA14] or manual labeling [VEL18a]. Finally, since the lidar shall be used in this work to provide ground-truth for object boundaries, the detection of the ground plane has to be adapted to the perceptive capabilities of the used radars. Here, to the best of the authors knowledge, only height threshold-based ground plane removal has been proposed [WES19, SLE19].

After distinguishing the ground plane detections, several possibilities to adapt the ISM arise. The first possibility is to adapt the IDMs for both the ground and object detections. Here, the ground point IDM must only apply the free space part of the model. For the object IDM, the ground detections shall be ignored so that they allow the casted rays to pass through and not to cause any collisions. The problem with this approach is that a ray needs to be cast for every detection which, for spinning lidars with 32 beams and more, can only be handled by discretizing the detections into a BEV image (e.g. Velodyne's HDL-32E provides up to 1.39 million points/second [21]). Therefore, this method is rarely adopted in practice.

Another alternative is to remove the ground points and only apply the IDM for object detections. However, by removing the ground plane detections, the information about free space gets lost for areas not affected by the object detection's rays, as illustrated in the left sector in Fig. 2-3 b). To solve this problem under the assumption of dense detections, IDM rays are cast for each angle up to a maximum distance within the lidars Field of View (FoV) instead of only for angles with detections. In doing so, the IDM can still be applied for all detections hit by rays and all the rays ending at maximum distance are altered to only model the free space. This variant is favored by the majority of works [THR06, NAR18, FIS81, HOU62, OLI16, TIA20].

In case the density assumption is violated, this approach, however, might lead to casting free space rays in between object detections assigning actually occupied space as partially free (see Fig. 2-3 b) right sector). To counteract this effect, it is possible to adapt the BEV discretization according to the point clouds density to close the gaps between detections or to increase the opening angle of the IDM rays to adapt for increased point cloud sparsity with increasing distance. The effects of counteracting the density violation are illustrated in Fig. 2-3 c).

2.3 Deep inverse Sensor Models

In recent years, the literature solely focuses on the creation of deep ISMs, which utilize deep Convolutional Neural Network (CNN)s to learn the characteristics of ISMs.

2.3.1 Deep ISM Architecture

Architecture-wise, the majority of works utilize UNets [RON15] consisting of an encoder and a subsequent decoder network with skip connections [PRO19, SLE19, WIR18, WES19, SCH18, LU19, MAN20]. The encoder network successively subsamples, commonly by a factor of two, the feature dimension either using a form of pooling (e.g. max or average pooling) to obtain shift invariance or strided convolutions as an alternative with a learned kernel. This results in bringing the spatial dimensions closer allowing the computation of increasingly global features while keeping the convolution kernel size small. In the standard setup, the padding is set to $(\kappa - 1)/2$ in order to pad the kernel overlap at the borders. Regarding the kernel size, only 3×3 convolutions are applied since it has been shown in [SIM14] that two 3×3 convolutions result in the same receptive field as one 5×5 convolution but need less parameters. As a downsampling mechanism, most papers deploy strided convolutions with stride size two as it is a learnable operations as opposed to the pooling alternatives. To define the depth of the network, the resulting receptive field size is essential and has to be chosen specifically for the task at hand. For the described standard operations with kernel size $\kappa = 3$ and stride $s \in [1, 2]$ and padding of one, the receptive field ($\mathrm{RecepField}$) can be computed as follows

$$\mathrm{RecepField}_{i+1} = \mathrm{RecepField}_i + (\kappa_i - 1) \prod_{j=0}^{i} s_j \qquad \text{Eq. 2-15}$$

The decoder is the inverse of the encoder using a bilinear upsampling [ODE16] to replace the subsampling layers. The skip connections either add or concatenate information of the encoder to corresponding decoding layers. This is done in order to regain the information lost in or during encoding. These skip connections can include additional convolutions which are mostly used to compress the amount of features before concatenation. While the feature dimension is successively halved in the encoder, the amount of features is commonly doubled after each subsampling. This introduces the initial amount of features as an architectural hyperparameter and, in some cases, a maximal number of features. The architecture search is being standardized in [RAD20] for a generalized UNet architecture with ResNet layers by successively evaluating the influence and ranges of each parameter. An example of such a standard UNet architecture is depicted in Fig. 3-7. In most cases, the convolution (conv) layers consist of 3×3 or 1×1 convolutions, in some cases with Dropout [SRI14] or stride s, followed by

batch normalization (BatchNorm) [IOF15] and a non-linearity like leaky Rectified Linear Unit (ReLU) [MAA13], which is depicted on the left-hand side in Fig. 2-5.

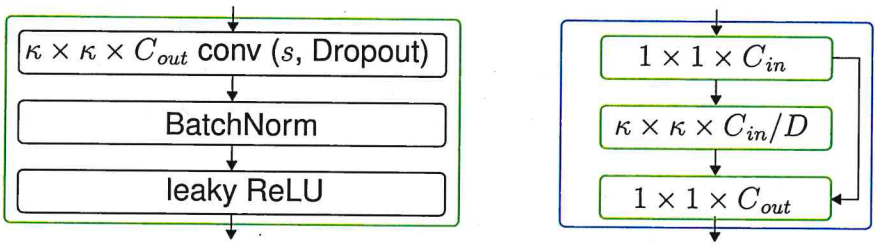

Fig. 2-5: Structure of a convolution (left) and ResNet layer (right) as commonly used in the literature with the kernel size κ, stride s, amount of channels C and channel reduction factor D. The green blocks in the ResNet layer are convolutions as depicted on the left-hand side.

Some of the variations to UNets include exchanging the encoder with backbone networks like VGG19 [SIM14, WUL18] or EfficientNet [TAN19, PHI20]. However, most commonly, the standard convolution layer is replaced by a ResNet layer [HE16, WIR18, REI20, ROD20, PHI20]. Here, the 3×3 convolution is replaced by a miniature UNet first compressing the feature channels with a 1×1 convolution, followed by the actual 3×3 convolution and, finally, decompressing it back to the original channel dimension, again, using a 1×1 convolution. Additionally, before applying the final output non-linearity, the input features are added to the outputs. The sum of Multiply-Accumulate Operations (MAC) over the ResNet layer's three convolutions can be written as follows

$$\text{Summe} \; ? \; = \; 1 \cdot 1 \cdot WHC_{in}\frac{C_{in}}{D} + 3 \cdot 3 \cdot WH\frac{C_{in}^2}{D^2} + 1 \cdot 1 \cdot WH\frac{C_{in}}{D}C_{out} \qquad \text{Eq. 2-16}$$

for features of width W, height H, a channel reduction by D and C_{in}, C_{out} input and output channels respectively. For $C_{in} = C_{out}$, the ResNet layer needs less MACs compared to the $9 \cdot WHC_{in}^2$ MACs of a 3×3 convolution for $D > 1.12$. Different sources report only minor accuracy decrease for $D = 4$ or higher, leading to 11.8% of the MACs needed for the standard convolution [HE16, BAZ18].

While all authors agree upon the fact that re-weighting the loss to account for class imbalance is necessary, they diverge in the choice of the underlying loss function. In the majority of cases, the training is setup as a semantic segmentation problem, training the network on the cross entropy loss [PRO19, LOM17, HEN20, WUL18]. On a different note, Weston et al. [WES19] and Lu et al. [LU19] train a Variational Autoencoder (VAE) [KIN13] for which the log-likelihood is optimized in visible and the Kullback-Leibler divergence [KUL51] in occluded areas. This results in a Gaussian distributed log-odds space which equals the prior's distribution in unobserved areas. Weston et al. further define the log-odds as priors to the final sigmoid activation and

arrive at the final prediction by marginalizing over the log-odds. Another alternative is proposed by Sless et al. [SLE19] who apply the Lovasz loss [BER18] as a differentiable surrogate to the intersection-over-union. Here, the Lovasz loss comes with the benefit of accounting for class imbalances by design. Eventually, Wirges et al. [WIR18] define the training as a regression problem and, thus, apply the $\mathcal{L}_1$ and $\mathcal{L}_2$ losses for optimization. This enables them to utilize the continuous nature of the training targets to continuously dampen the influence of the loss in areas with high target uncertainties.

When it comes to choosing an optimizer, the vast majority of works use the ADAM optimizer [KIN14, VER19, WIR18, WES19, SCH18].

2.3.2 Deep ISMs for Cameras

In the vision domain, the environment representation is often elevated from the binary classification into free and occupied space towards estimating a more detailed semantic layout including classes like vehicles, streets, sidewalks, vegetation etc. Since there are a vast amount of different approaches to tackle the problems in this field of research, Fig. 2-6 provides a short overview of the solution space. The main problem in training a deep Inverse Camera Model (ICM) to estimate such a semantic layout is the transformation between camera and BEV projection. To solve this, Schulter et al. [SCH18] propose to preprocess the images by estimating the monocular depth and semantic information. These estimates are further used to create a semantically annotated BEV image which is then fed into a neural network for refinement. In [PHI20], an end-to-end architecture is proposed to combine the Monocular Depth Estimation (MonoDepth) estimation and the BEV refinement into a single network. This is achieved by first predicting a feature vector and a categorical depth distribution for each pixel, resulting in a 3D point cloud in the shape of a pyramid. Each 3D point is then assigned with the feature vector, scaled by the probability of the categorical depth distribution. Here, the learned feature vector has the potential to encompass more relevant information than the semantic labels provided in [SCH18]. Afterwards, the point cloud is projected into BEV and further refined by another, simultaneously trained network. An alternative path to handle the transformation to BEV is to directly feed the image into a Fully Convolutional Network (FCN) and learn the transformation implicitly, as proposed in [MAN20, LU19]. To obtain a more explicit integration of the transformation into the network, Reiher et al. [REI20] and Roddick and Cipolla [ROD20] proposed use a spatial transformer layer [JAD15] to transform the latent space according to the homography defined by the camera intrinsics and exterinsics as explained in Section 2.2.3. Similarly, Pan et al. [PAN20] define a custom layer called "View Transformer Module" which uses a Multilayer Perceptron (MLP) to learn the positional mapping between the compressed camera and the BEV features.

supervision signal	BEV projection			
	MonoDepth + SemSeg	FCN	spatial trafo. layer	learned trafo. layer
adversarial	[SCH18]	[MAN20]		
maps	[SCH18, PHI20]	[MAN20]	[ROD20]	
sparse supervision		[MAN20, LU19]		
simulation		[REI20]	[REI20]	[PAN20]

Fig. 2-6: Overview of deep ICM literature with row-wise supervision approaches and column-wise solutions for the transformation between camera and BEV projection

2.3.3 Deep ISMs for Range Sensors

The origins of approximating ISMs for range sensors using neural networks go back to the 90s [VAN95, THR93]. One this range sensors have in common is the fact that the measurement signals can be directly projected into BEV images and fed into neural networks. This makes the training on radar and lidar data quite similar. One differentiating factor, however, is the measurement signal representation. In case of lidar, the simplest representation can be obtained by projecting the detections into BEV and feeding it into the neural network [LIA18]. To provide additional input information, Wirges et al. [WIR18] use the intensities, detection positions and transmissions measured by the lidar and mark them in separate channels each for ground and non-ground points, arriving at six input channels. Hendy et al. [HEN20] additionally compute the density of all lidar detections and the maximal height value for each grid cells as input features, resulting in eight channels. Eventually, Wulff et al. [WUL18] use the detection positions, cell density, height thresholded detections, six height statistics (min, max, mean, min-max difference, mean-standard-deviation, mean-variance) and the same six statistics for the reflectivity which sums up to a 15 channel input. Similar approaches can be observed when it comes to radar sensor. Here, it has been proposed to either use the detection positions directly [SLE19] or to accumulate detections over time in order to account for the sparsity [PRO19, LOM17]. Additionally, features like radar cross section, signal to noise ratio, ambiguous Doppler interval and relative x and y velocities can be encoded into separate channels [HEN20]. Weston et al. [WES19] use a radar setup which provides the raw, dense range returns without Doppler information. These measurements are projected into a polar BEV image and fed into the network. The only other approach that uses polar instead of Cartesian coordinates is proposed by Verdoja et al. [VER19]. However, instead of encoding the inputs and targets as images, they rather perform inference on a vector of range measurements where each dimension

To show this property, it is sufficient to prove it for one of the in-going unknown masses, since Dempster's combination rule is associative. Thus, the property shall be shown under the assumption that m_{u1} is the bigger in-going unknown mass. It then follows that

$$m_{u1} \geq m_{u12} \underbrace{=}_{\text{with Eq. 2-18}} m_{u1}\frac{m_{u2}}{1-K} \quad |\div m_{u1} \qquad \text{Eq. 2-20}$$

In case of $m_{u1} = 0$ both sides equal zero. For $m_{u1} \in (0, 1]$, the following holds

$$1 \geq \frac{m_{u2}}{1-K} \underbrace{=}_{\text{with Eq. 2-5}} \frac{1-m_{f2}-m_{o2}}{1-K} \quad |\cdot(1-K) \qquad \text{Eq. 2-21}$$

$$1-K \underbrace{=}_{\text{with Eq. 2-17}} 1-m_{f1}m_{o2}-m_{o1}m_{f2} \geq 1-m_{f2}-m_{o2} \quad |-1+m_{f1}m_{o2}+m_{o1}m_{f2} \qquad \text{Eq. 2-22}$$

$$0 \geq m_{f2}\underbrace{(m_{o1}-1)}_{\leq 0} + m_{o2}\underbrace{(m_{f1}-1)}_{\leq 0} \qquad \text{Eq. 2-23}$$

The alternative to Dempster's rule, which is equally often applied in evidential occupancy mapping [WIR18, KUR12, REI13], is Yager's rule of combination [YAG87] as shown for the occupancy specific case by the following equations

$$\boldsymbol{m}_1 \oplus_Y \boldsymbol{m}_2 = \begin{bmatrix} m_{f1}m_{f2} + m_{f1}m_{u2} + m_{u1}m_{f2} \\ m_{o1}m_{o2} + m_{o1}m_{u2} + m_{u1}m_{o2} \\ m_{u1}m_{u2} + K \end{bmatrix} \qquad \text{Eq. 2-24}$$

In contrast to Dempster's rule, Yager's rule redistributes the conflicting portion of the fused masses into the unknown class. In the case of big conflicts K, this allows to recuperate unknown mass.

For the sake of completeness, it shall be mentioned that, for the general, multi-hypothesis case, there is a big body of literature describing the shortcomings of Dempster's and Yager's rule together with propositions of how to handle them [ZAD79, HAN08, YAN13, ZHA20b].

2.4.2 Combination of dependent Evidence

In case of dependencies between evidences, direct combination, as proposed is Section 2.4.1, leads to accounting twice for the dependent portion of information. The literature proposes two lines of solutions namely removing the redundancy of one of the evidence masses before combining them or adapting the combination rule to account for occurring redundancies. Here, the problem with newly proposed combination