ROI-Aware Neural Image Compression for Teleoperation

Oriol Gorriz Viudez
Institute of Automotive Technology
Technical University of Munich (TUM)
Garching bei München, Germany
oriol.gorriz@tum.de

***Abstract*—Teleoperated driving relies on real-time image transmission, demanding efficient compression strategies that preserve critical scene details while minimizing bandwidth usage. Traditional codecs (e.g., HEVC, AVC) apply uniform compression and lack support for variable bitrate for regions of interest (ROIs). To address this, we propose a fully neural image compression framework optimized for images captured in autonomous driving environments. Our approach integrates an object detector with a neural compressor, conditioning the compression process on object-specific features extracted from the detector. By incorporating these features as additional input to the compressor and employing a custom loss function that prioritizes ROIs, our method achieves enhanced preservation of important scene elements while reducing bitrate allocation to less relevant background areas. Experiments show that our method improves compression efficiency and visual quality in ROIs of teleoperated driving images.**

***Index Terms*—autonomous driving, teleoperated driving, image compression, object detection**

I. Introduction

A. Motivation

In a realistic, fully autonomous driving scenario, especially considering external factors or system malfunctions where safety cannot be guaranteed, a remote operator (RO) must be able to take control of the vehicle. To maximize the likelihood of successful teleoperation, a low-latency, high-resolution video stream should be provided to the RO.

Traditional video codecs, such as AVC [1] and HEVC [2], are widely used for teleoperation. While they do not offer the most advanced compression efficiency, they provide a favorable balance between compression performance and computational efficiency [3], making them well-suited for general consumer applications. However, they lack adaptability to scene content, leading to unnecessary bandwidth use or loss of critical details since the compression is uniform [4].

On the other hand, neural compression is a new alternative that allows better compression efficiency at the expense of larger computation requirements [3], [5], but they also offer interesting capabilities such as a direct integration with other neural networks that are already present in the autonomous system software stack. This integration can benefit the neural compressor by providing it with image features that are already being used in other components and that may be used to enhance the overall performance of the neural codec.

In the context of teleoperated driving, considering the high resolution of nowadays sensors, an often low-speed internet connection to the vehicle or the possibility of multiple ROs connecting to the vehicle, naively compressing the video stream can lead to a low resolution stream as bandwidth is uniformly allocated.

We observe that not all image regions are equally important for the RO in the dynamic driving task (DDT). Certain regions, such as those containing high-frequency information like traffic signs or moving objects, are significantly more critical for accurate task performance. In contrast, areas depicting the sky, buildings, or uniform backgrounds tend to carry less importance.

This highlights the need for dynamic compression, focusing on ROIs where the most relevant information for teleoperation is located. We can implement a neural codec where the objective is not to minimize the overall rate distortion but to allow a learnable differentiation in compression between ROIs and non-ROIs. In addition, and to enhance the ROI differentation, we can make use of the semantic information that is already present in the autonomous vehicle software stack.

B. Overview

In this paper, we explore the advantages of neural image compression over traditional codecs, particularly the ability to condition the encoding process on environmental and object-level features already available in an autonomous driving software stack.

We introduce a neural compression framework that dynamically adjusts the visual quality by allocating more bits into ROIs conditioned on object features, allowing for higher visual quality in ROIs while maintaining overall compression efficiency.

Our method is trained and evaluated on the KITTI 2D object detection dataset [6], which provides representative imagery for real-world autonomous driving scenarios. In this work, we consider every labeled object in the dataset as an object of interest (OOI), and we define ROI as all pixels that lie inside an OOI's bounding box.

To validate our approach, we compare it against a baseline that does not leverage object features and evaluate its generalization performance across different domains.

C. Contributions

Our work introduces a neural image compression framework that dynamically adjusts compression efficiency based on object features extracted from an object detector. Specifically, we propose:

- A neural image compression framework that conditions compression efficiency on object features provided by a lightweight YOLOv8-nano [7] object detector.
- A modification of the variational scale hyperprior image compressor proposed by Ballé et al. [8] that extends the model by adding another prior for the object features and integrating it into the compression formulation.
- An adjusted rate distortion train objective that separates the global image distortion into two distortion components, one for ROIs and another for non-ROIs. We get ROI information directly from ground-truth (GT) objects. A parameter α allows to set a tradeoff between the visual quality (distortion) of ROIs and non-ROIs.

II. Related Work

Neural image codecs (NICs), also known as learned image codecs (LICs), have advanced significantly in recent years. Among the most influential contributions are those by J. Ballé et al. [8]–[10]. In particular, the introduction of the scale hyperprior model [8] represents a foundational breakthrough that enabled NICs to become competitive with, and often superior to, traditional consumer-oriented codecs such as JPEG. A comprehensive comparison is provided in [11]. The impact of these developments has been substantial enough that the Joint Photographic Experts Group (JPEG) is now developing a standard based on neural networks, known as JPEG AI [12].

Furthermore, neural video codecs (NVCs), an extension to NICs, have also gotten significantly better than traditional codecs [3], with the most prominent approaches coming from Microsoft Research and its Deep Contextual Video Compression (DCVC) series [5], [13]–[17], that achieve state-of-the-art performance on benchmarks like UVG [18]. These approaches aim to model both spatial and temporal redundancy in video sequences. While traditional codecs rely heavily on motion estimation and compensation, end-to-end DCVC approaches rely on learned context models that directly predict and compress residuals between frames leveraging neural networks and optimizing a rate-distortion loss. Their most recent proposal, DCVC-RT [17], achieves 110 fps coding on 1080p video with a 21% bitrate savings compared to H.266/VTM [19].

ROI-based compression codecs, are a less active field of research. Approaches like [20] from Wang et al. propose a method to allocate more bitrate in those areas that they consider important (e.g., road, object, human). A Semantic-Aware Compression (SAC) framework binarily differentiates between ROIs and non-ROIs within video frames, and compresses them separately using different bitrates with standard codecs to enhance the quality of critical areas.

Cai et al. proposes an end-to-end method for ROI compression [21] that includes a fully-differentiable ROI prediction scheme. In addition, they propose an adapted rate-distortion loss function that factors-in the segmentation error and that includes two distortion components, one weights the ROI distortion and the other weights the non-ROI distortion. Even though this method is end-to-end it still makes a clear binary differentiation between ROIs and non-ROIs and is essentially a segmentation neural network sequentially connected to a neural compressor, we still see room for improvement.

Jiang et al. [22] recently introduced a method that leverages a sparse LiDAR point cloud to enhance the image compression performance. Their method demonstrates that adding more features correlated to the input image can improve the overall compression efficiency, effectively conditioning the compression on those features and reducing the entropy. They propose to first project the point cloud to a 2D plane, then generate features from the depth map, and finally combine them with those of the image and give them as input to the entropy model. Particularly remarkable from this approach is how they condition the compression using features from two different sources.

NIC and NVC are very active fields of research, our method does not intend to achieve a state-of-the-art compression efficiency, however, we show that by conditioning the compression on object features, the overall efficiency is higher. Our framework could theoretically be applied to all NICs with architecture modifications, however, the impact of it is out of the scope of this work.

III. Methodology

In this section, we introduce our image compression framework that leverages object features to enhance image compression efficiency and allow for dynamic ROI distinction. We discuss the difficulties we have faced and the key design choices that we have made. We also introduce the training setup alongside with the evaluation methods.

Our method consists of an adjusted rate-distortion objective function and the incorporation of object features into the variational posterior of the compressor. While we demonstrate its integration with a specific object detector and compressor architecture, the approach is *general* and not restricted to these particular choices. Due to our concept being model-agnostic, it could also be applied to neural video compression.

A. Notation

We define $\boldsymbol{x} \in \boldsymbol{X}$ as the input image, $\boldsymbol{y} \in \boldsymbol{Y}$ as the latent representation of the image, $\boldsymbol{f} \in \boldsymbol{F}$ as the object features, $\boldsymbol{z} \in \boldsymbol{Z}$ as a latent variable used to describe uncertainty (scale hyperprior), $\boldsymbol{w} \in \boldsymbol{W}$ as a latent variable used to describe the uncertainty of the object features, $\boldsymbol{\phi}$ as neural encoder parameters, and $\boldsymbol{\theta}$ as neural decoder parameters.

We define the notation $\hat{\boldsymbol{y}}$ to denote discrete quantization, while $\tilde{\boldsymbol{y}}$ represents the addition of uniform noise, which serves as a differentiable approximation to quantization during training. The reconstructed image is $\hat{\boldsymbol{x}}$.

B. Architecture Overview

Our method intends to inform the image compressor by providing it with additional semantic information about the image content, i.e., where the objects are in the image, their size, and their importance. There are two main components, the object detector and the compressor model.

Regarding the object detector, we obtain the object features from 3 forward hooks at different relevant layers of a YOLOv8-nano [7] object detector that is already trained and that is fine-tuned on the KITTI 2D object detection dataset [6] by Dani [23]. This object detector, with 3.5 million parameters, is trained to detect the following object classes: *Car, Pedestrian, Van, Cyclist, Truck, Misc, Tram, Person_Sitting*. We consider an object of any of these classes an object of interest (OOI) and we consider all pixels lying inside an OOI's bounding box a ROI.

As for the compression model, we extend and take as a baseline the variational image compressor with a scale hyperprior proposed by Ballé et al. in 2018 [8]. This model is essentially a variational autoencoder (VAE) that minimizes the total expected code length by learning to balance the amount of side information with the expected improvement of the entropy model. Specifically, they introduce the concept of side information that can be viewed as a prior on the parameters of the entropy model, making them hyperpriors of the latent representation.

In variational inference, the goal is to approximate the true posterior distribution $p_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})$, which is assumed to be intractable, therefore it is approximated with a parametric variational density $q(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})$. By leveraging the concept of the hyperprior [8], this becomes $q(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}} \mid \boldsymbol{x}, \boldsymbol{\phi})$, and is defined as a single joint factorized variational posterior, defined in (1).

$$q(\tilde{y}, \tilde{z} \mid x, \phi_g, \phi_h) = \prod_i \mathcal{U}\Big(\tilde{y}_i \mid y_i - \frac{1}{2},\, y_i + \frac{1}{2}\Big) \cdot \prod_j \mathcal{U}\Big(\tilde{z}_j \mid z_j - \frac{1}{2},\, z_j + \frac{1}{2}\Big) \quad (1)$$

$$\text{with } \boldsymbol{y} = g_a(\boldsymbol{x}; \phi_g), \boldsymbol{z} = h_a(\boldsymbol{y}; \phi_h),$$

where $\mathcal{U}(a, b)$ denotes a uniform distribution between a and b and is a stand-in for quantization.

We extend the hyperprior concept, as shown in Fig. 1, with an object prior, this new prior is intended to model the uncertainty of the latent representation of the object features $\boldsymbol{w}$. The underlying intuition, depicted in (2), is that by conditioning on more, informative data, the overall entropy needed to encode the image is expected to decrease.

$$H(\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{z}}) \geq H(\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{z}}, \hat{\boldsymbol{w}}) \quad (2)$$

Our extended variational posterior is $q(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{w}} \mid x, f)$, and is defined in (3).

$$q(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{w}} \mid \boldsymbol{x}, \boldsymbol{f}; \boldsymbol{\phi}) = \prod_i \mathcal{U}\left(\tilde{y}_i \mid y_i - \frac{1}{2}, y_i + \frac{1}{2}\right) \cdot \prod_j \mathcal{U}\left(\tilde{z}_j \mid z_j - \frac{1}{2}, z_j + \frac{1}{2}\right) \cdot \prod_k \mathcal{U}\left(\tilde{w}_k \mid w_k - \frac{1}{2}, w_k + \frac{1}{2}\right) \quad (3)$$

$$\text{with } \boldsymbol{y} = g_a(\boldsymbol{x}, \boldsymbol{f}; \phi_g), \boldsymbol{z} = h_a(\boldsymbol{y}; \phi_h), \boldsymbol{w} = i_a([\boldsymbol{y}; \boldsymbol{f}]; \phi_i),$$

where $[\cdot\,;\cdot]$ represents a channel-wise concatenation of two tensors.

Our prior is now $p_{\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{z}},\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{y}} \mid \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{w}})$ and the expected code length becomes:

$$\mathbb{E}_{x\sim p_x, f\sim p_f}\left[-\log p_{\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{z}},\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{y}} \mid \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{w}})\right] \quad (4)$$

We obtain the scaler of $\tilde{\boldsymbol{y}}$ by decoding the two latent variables $\hat{\boldsymbol{z}}$ and $\hat{\boldsymbol{w}}$. We then concatenate the two scale corrections $\hat{\boldsymbol{\sigma}}_{\boldsymbol{z}}$ and $\hat{\boldsymbol{\sigma}}_{\boldsymbol{w}}$ to obtain the scaler $\hat{\boldsymbol{\sigma}}$. Specifically as described in (5).

$$\hat{\boldsymbol{\sigma}} = [\hat{\boldsymbol{\sigma}}_{\boldsymbol{z}}; \hat{\boldsymbol{\sigma}}_{\boldsymbol{w}}] \quad (5)$$

$$\text{with } \hat{\boldsymbol{\sigma}}_{\boldsymbol{z}} = h_s(\hat{\boldsymbol{z}}; \theta_h), \hat{\boldsymbol{\sigma}}_{\boldsymbol{w}} = i_s(\hat{\boldsymbol{w}}; \theta_i).$$

Finally, (6) defines the conditional prior for $\tilde{\boldsymbol{y}}$.

$$p_{\tilde{y}|\hat{z},\hat{w}}(\tilde{y} \mid \hat{z}, \hat{w}, \theta) = \prod_i \left(\mathcal{N}\big(0, \hat{\sigma}_i^2\big) * \mathcal{U}\big(-\tfrac{1}{2}, \tfrac{1}{2}\big)\right)(\tilde{y}_i), \quad (6)$$

where the notation '*' denotes convolution, $\mathcal{N}(\mu, \sigma^2)$ is a Normal distribution with mean μ and variance σ^2.

It is worth highlighting how object features are integrated with the input image, as illustrated in Fig. 1. Following a merge strategy similar to that of [22], we interleave the fusion of multi-scale object features with convolutional and GDN [9] layers. At each stage, an object feature map is first projected using a 1×1 convolution and upsampled to match the spatial dimensions of the current $\boldsymbol{x}$ tensor. This projection is then concatenated with $\boldsymbol{x}$, after which the next convolution and GDN [9] are applied. This interleaving continues for each level of object features, enabling a gradual and spatially-aligned integration into the encoding path.

C. Adjusted Rate Distortion Objective

In neural image compression, the goal is to balance the trade-off between image quality (distortion) and compression efficiency (rate). This is typically achieved by optimizing a rate-distortion (RD) loss function, which combines a rate term (measuring the number of bits required to represent the compressed image) and a distortion term (measuring how much the reconstructed image deviates from the original). Mathematically, the RD loss is formulated as a Lagrangian optimization problem, defined in (7).

$$\mathcal{L}_{\text{RD}} = R(\hat{\boldsymbol{y}}) + \lambda D(\boldsymbol{x}, \hat{\boldsymbol{x}}), \quad (7)$$

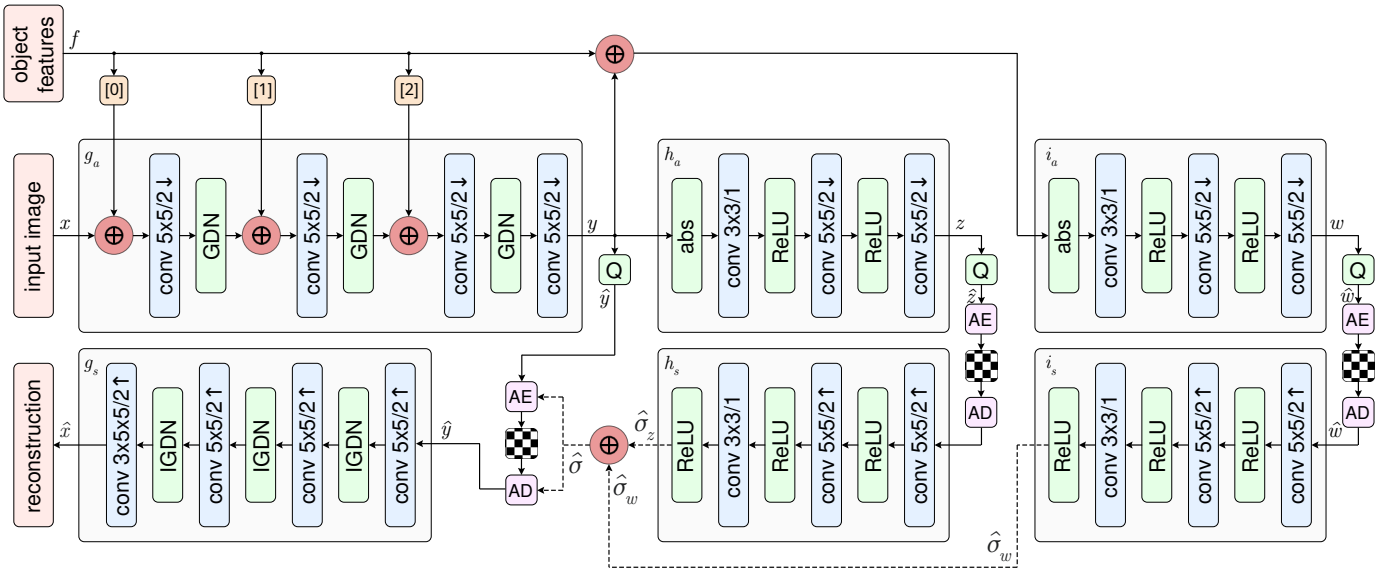

Fig. 1. Network architecture of our neural image compressor, an extension of [8]. The left side shows an image autoencoder architecture that includes object features, on the middle and right there are two hyperpriors, one that models the sigmas corresponding to the input image and the other that models the sigmas corresponding to the object features. Q represents quantization, AE and AD represent arithmetic encoder and arithmetic decoder, respectively. $\oplus$ block represents a channelwise concatenation. Convolution parameters are denoted as: number of filter $\times$ kernel support height $\times$ kernel support width / down- or upsampling stride, where $\uparrow$ indicates upsampling and $\downarrow$ downsampling. GDN/IGDN blocks are non-linear transforms from [9].

where $R(\hat{\boldsymbol{y}})$ is the amount of bits needed, $D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is the distortion (e.g., MSE), and λ is the Lagrange multiplier that balances the importance of rate vs. distortion.

This formulation allows the model to jointly learn how to minimize the reconstruction error while also learning an efficient representation of the latent space suitable for entropy coding.

Nevertheless, we extend the traditional rate-distortion loss (7) to enable the VAE to support variable compression, preserving higher quality in ROIs while allowing stronger compression in non-ROI regions, at the cost of increased distortion in those areas. To achieve this, we decompose the distortion term into two components: one corresponding to ROIs and another to non-ROIs. We introduce a weighting parameter α to control the trade-off between the quality of ROIs and non-ROIs. We obtain ROI information directly from the GT, decoupling the VAE's training objective from the results of our object detector, this makes the training less reliant on the confidence threshold parameter of the YOLO detector since the features describe the overall scene. Additionally, each distortion term is normalized by the proportion of pixels in its respective region (ROI or non-ROI) relative to the total number of pixels in the image. The modified rate-distortion loss is shown in (8).

$$\mathcal{L}_{\text{RD}} = R(\hat{\boldsymbol{y}}) + \lambda(\alpha \cdot D_{\text{ROI}} + (1-\alpha) \cdot D_{\text{non-ROI}}) \tag{8}$$

where $D_{\text{ROI}} = \frac{N_{\text{ROI}}}{N} D(\boldsymbol{x}_{\text{ROI}}, \hat{\boldsymbol{x}}_{\text{ROI}})$ is the normalized distortion of all pixels belonging to a ROI, $D_{\text{non-ROI}} = \frac{N_{\text{non-ROI}}}{N} D(\boldsymbol{x}_{\text{non-ROI}}, \hat{\boldsymbol{x}}_{\text{non-ROI}})$ is the normalized distortion of all pixels not belonging to a ROI and $\{\boldsymbol{x}_{\text{ROI}}, \boldsymbol{x}_{\text{non-ROI}}\}$ are obtained by masking $\boldsymbol{x}$ with the GT object bounding boxes.

The effect that the parameter α has on the overall distortion component is:

- $\alpha \approx 1$ implies that the overall distortion component will mostly be D_{ROI}, allowing the VAE to train on only ROIs.
- $\alpha = 0.5$ implies that the same weight is given to ROIs and non-ROIs. It behaves exactly the same as with using (7).
- $\alpha \approx 0$ implies that the overall distortion component will mostly be $D_{\text{non-ROI}}$, allowing the VAE to train on only ROIs.

D. Training Setup

We select the KITTI 2D object detection dataset [6] as our source of imagery and object labels during training. It is comprised of 7481 labeled images, we split it into training and test sets with 95% and 5% of samples, respectively.

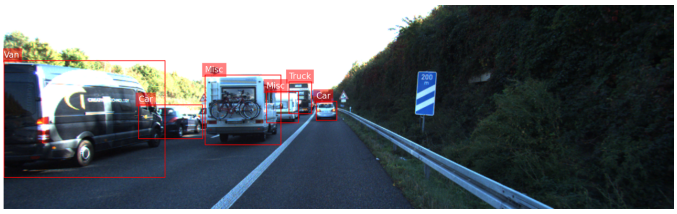

Fig. 2. Labeled sample of the KITTI 2D object detection dataset.

Fig. 2 shows a labeled sample of the dataset. The size of each sample is 1242×375 pixels. Our object detector, a YOLOv8-nano [7], expects squared images of 640×640 pixels. Our image compressor is not bound to any resolution, but it should receive the same visual content as the object detector.

We first randomly extract the largest possible square crop from the input image. This crop is then duplicated and resized into two versions: one of size 640×640 pixels, used as input to the object detector, and another of size 256×256 pixels, denoted as $\boldsymbol{x}$, used as input to the image compressor. For any object whose bounding box intersects the crop boundary, we adjust the bounding box by cropping it accordingly, only the visible portion within the square crop is retained as the new bounding box.

Our training optimization function is (8), and we select the mean squared error (MSE) as the distortion metric. We use the Adam optimizer alongside the OneCycleLR [24] learning rate scheduler.

We have taken the baseline model [8] implementation from CompressAI [25], and extended it with our method as described in III-B.

We only train the compressor model—since the object detector is already fine-tuned on this dataset—on a Nvidia L40 46GB. We use PyTorch as our deep learning library of choice, and let it train for 300 epochs, with a batch size of 128 and a maximum learning rate of 6.0e-4. Each epoch takes $\approx$ 1 minute, for a total training time of $\approx$ 5 hours.

Due to the fact that our selected NIC does not allow to change the compression quality once it has been trained, we need to retrain the model each time we want a different rate-distortion ratio (λ parameter) or we want to change the weight given to ROIs or non-ROIs (α parameter).

IV. Results

Here we show the results of our method compared against our baseline, both described in III-B.

We measure the performance using the Peak Signal-to-Noise Ratio (PSNR). With the purpose of showing more than one point in the PSNR plots, and since our compressor does not allow to change the compression rate during inference, for every set of parameters, we train multiple models with different λ.

We show PSNR plots focusing on ROIs and plots focusing on non-ROIs. Equation (9) shows how we compute the PSNR values, PSNR_l, for $l \in \{\text{ROI}, \text{non-ROI}\}$.

$$\mathrm{PSNR}_l(\boldsymbol{x}_l, \hat{\boldsymbol{x}}_l) = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}(\boldsymbol{x}_l, \hat{\boldsymbol{x}}_l)}\right) \tag{9}$$

A. An Improvement Over the Baseline

Here we compare the performance of the baseline model against our proposal that includes object features, both trained by us. To do so, we have trained our method with a high emphasis on ROIs by setting $\alpha = 0.995$. Fig. 3 shows the PSNR values obtained in ROIs and measured at three different λ values. Contrary to our expectations, Fig. 3 shows that with the same three λ values, we get an overall lower PSNR using our method. This could be attributed to the different behaviour of the rate-distortion objectives (7) and (8), setting the same λ values for both approaches does not guarantee the same rate nor distortion, making it harder to do a fair comparison.

On the other hand, point (a) from the baseline and point (c) from our method, taken with $\lambda = 960$ and $\lambda = 29760$ respectively, achieve similar bits per pixel values but our method achieves has a higher PSNR value: 5.43% higher PSNR while using only 3.88% more bits per pixel. Equally important, point (b) from our method achieves a higher PSNR value than point (a) from the baseline while taking less bits per pixel.

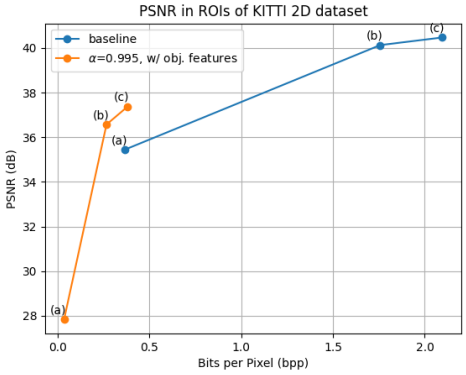

Fig. 3. PSNR in ROIs obtained by our baseline and our method with object features and $\alpha = 0.995$ on the KITTI 2D dataset. The points (a)-(c), are obtained with $\lambda = 960$, $\lambda = 15360$ and $\lambda = 29760$, respectively.

Looking at the PSNR values in non-ROIs, as seen in Fig. 4, our method achieves a lower PSNR compared to the baseline, indicating that our method is effectively transferring information from non-ROIs to ROIs.

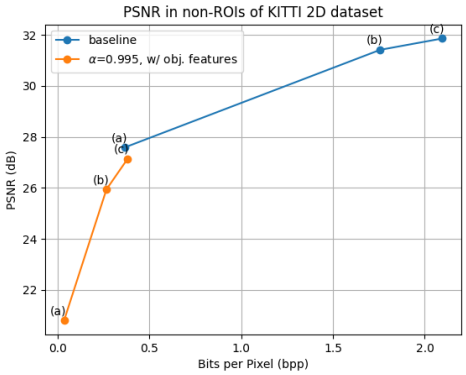

Fig. 4. PSNR in non-ROIs obtained by our baseline and our method with object features and $\alpha = 0.995$ on the KITTI 2D dataset. The points (a)-(c), are obtained with $\lambda = 960$, $\lambda = 15360$ and $\lambda = 29760$, respectively.

Interestingly, when comparing the magnitude of PSNR values between Fig. 3 and Fig. 4, we see that it is much lower in non-ROIs, indicating a higher compression of these areas even with our baseline and suggesting that more-uniform areas (non-ROIs) are compressed more aggressively to transfer bits to higher-frequency regions (ROIs) to achieve a lower rate-distortion loss. We also see an accentuation of this difference with our method, which is expected.

B. The Effect of Object Features

To delve deeper into what precisely is the impact of object features in our method, we will compare how our extended model, described in III-B, performs with and without object features. The object detector forward hooks are zero-filled for the test without object features, all the other parameters are exactly identical. We set $\alpha = 0.995$ for both models, we expect this way a much higher PSNR value in ROIs compared to non-ROIs.

Not completely satisfying our expectations, Fig. 5 shows a marginal but significant improvement in PSNR values when conditioning compression on object features. Specifically, point (c) shows that with object features, a 0.8% higher PSNR can be achieved while using 7.74% less bits per pixel. Fig. 6 shows an overall lower PSNR values in non-ROIs when compression is conditioned on object features, which is expected. Taking both figures into account, we realize that most of the PNSR improvement of our method is due to our modified training objective that prioritizes a low distortion in ROIs, defined in (8).

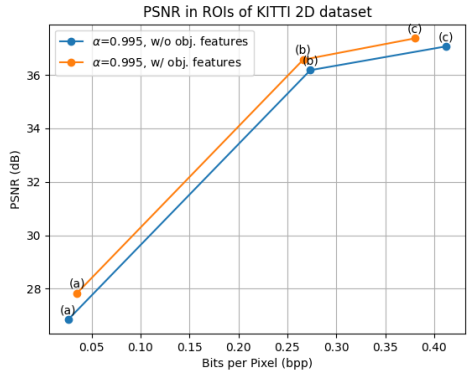

Fig. 5. PSNR in ROIs obtained by our method without and with object features, both with $\alpha = 0.995$ on the KITTI 2D dataset. The points (a)-(c), are obtained with $\lambda = 960$, $\lambda = 15360$ and $\lambda = 29760$, respectively.

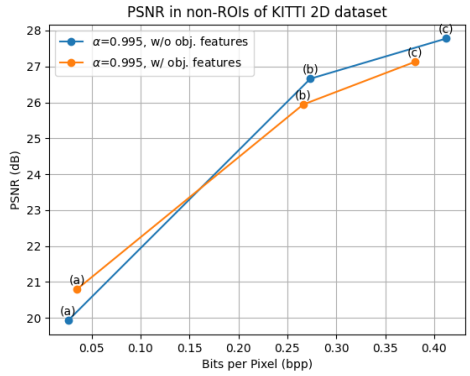

Fig. 6. PSNR in non-ROIs obtained by our method without and with object features, both with $\alpha = 0.995$ on the KITTI 2D dataset. The three points (a)-(c), are obtained with $\lambda = 960$, $\lambda = 15360$ and $\lambda = 29760$, respectively.

Fig. 7. Visual comparison of a ROI on the KITTI 2D dataset. (a) is the ground truth. (b) is obtained from the baseline with $lambda = 960$. (c) is obtained from our method without object features and with $\lambda = 29760$, $\alpha = 0.995$. (d) is obtained from our method with object features and with $\lambda = 29760$, $\alpha = 0.995$.

As reflected in Fig. 3 and Fig. 5, our method without object features outperforms the baseline in ROIs, and our method with object features achieves the highest PSNR among these. We show a visual comparison of a ROI on the KITTI 2D object detection dataset in Fig. 7. In this figure, we intend to visualize the difference in PSNR from point (c) of our method and point (a) of the baseline of Fig. 3. Notably, image (d) is crisper than the others when compared to the ground truth.

C. Performance in Another Domain

To further investigate the behaviour and generalization capabilities of our method, we evaluate it in a domain different from autonomous driving. We have selected the COCO 2017 image recognition dataset [26] to make this comparison. This is a generalistic dataset and contains images that do not lie in a specific domain.

However, before evaluating our method on the COCO dataset, we first examine how our baseline—trained on the KITTI 2D dataset—is affected by domain shift. The results are shown in Figs.8 and9 for ROIs and non-ROIs, respectively. For reference, we also include results from a pretrained baseline using the weights provided by CompressAI [25], also shown in the same figures. Surprisingly, the pretrained model outperforms our KITTI-trained baseline, despite being evaluated on this same dataset. We attribute this to a different training setup and access to broader or more diverse data. As expected, the performance gap between domains is larger for our KITTI-trained baseline, which has only been exposed to driving

scenarios—a much narrower distribution than that of COCO. It is also worth noting the difference in performance shift between ROIs and non-ROIs. Our baseline shows a significant performance drop in ROIs, but a nearly negligible shift in non-ROIs. This can likely be attributed to the fact that non-ROIs often contain low-frequency, textureless content, which is easier to compress and tends to generalize better across domains.

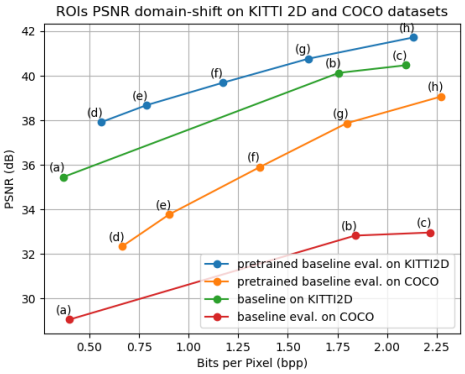

Fig. 8. ROI PSNR comparison of a pretrained baseline model by CompressAI [25] and our baseline model trained on KITTI 2D, both evaluated on KITTI 2D and COCO datasets. The points (a)-(c), are obtained with $\lambda = 960$, $\lambda = 15360$ and $\lambda = 29760$, respectively. The points (d)-(h) correspond to quality levels 4-8 of CompressAI.

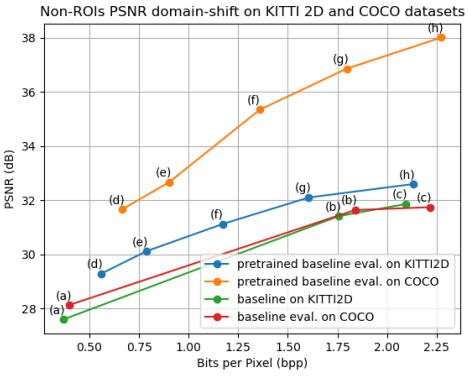

Fig. 9. ROI PSNR comparison of a pretrained baseline model by CompressAI [25] and our baseline model trained on KITTI 2D, both evaluated on KITTI 2D and COCO datasets. The points (a)-(c), are obtained with $\lambda = 960$, $\lambda = 15360$ and $\lambda = 29760$, respectively. The points (d)-(h) correspond to quality levels 4-8 of CompressAI.

We now evaluate how our method—trained on the KITTI 2D object dataset without object features and optimized for higher PSNR in ROIs—performs on the COCO 2017 image dataset without object features. As shown in Fig. 10, the results reveal a significant drop in ROI performance when applied to a different domain, with PSNR decreasing by 19.76% on average.

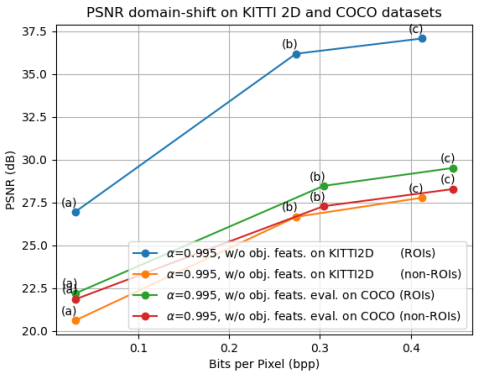

Fig. 10. PSNR of our method (trained on KITTI 2D dataset without object features and with a high emphasis on ROIs; $\alpha = 0.995$) and the same model evaluated on the COCO image dataset. The points (a)-(c), are obtained with $\lambda = 960$, $\lambda = 15360$ and $\lambda = 29760$, respectively.

V. Discussion

In this work, we propose a neural image compression method that explicitly prioritizes the preservation of detail in semantically important regions, or ROIs. Our experiments demonstrate that this approach leads to consistent improvements in PSNR within ROIs and a poorer performance in non-ROIs, effectively transferring information (bits) from non-ROIs to ROIs.

Our findings suggest that the modified training objective, which explicitly prioritizes low distortion in ROIs, plays a critical role in the observed performance gains. In contrast, conditioning the compression on object features extracted by an object detector leads to a more modest enhancement. Additionally, we demonstrate how our method generalizes to a broader domain, showing that the performance shift in non-ROIs when transitioning between domains is nearly negligible.

Some limitations must be acknowledged. First, although we decouple the overall compression to the detections of our object detector, the reliance on pre-defined or model-derived object features means that performance could vary depending on the accuracy of the detector on that specific domain. Second, we only tested our method on an extended version of the Scale Hyperprior (Fig. 1), the effect that our method has on other architectures is out of the scope of this work, even small variations of our architecture could drastically change results. Moreover, our experiments indicate that the current training setup can be significantly improved to achieve a higher baseline performance.

As future work, we plan to extend our method to video compression. Given that video compression is inherently an auto-regressive task, our approach remains fully applicable in this context. Several aspects of the current method could also be improved. Rather than relying on externally provided object features, the model could be modified to learn these features internally by mimicking or learning similar representations, completely separating the compressor from an object detector. Additionally, fine-tuning the object detector jointly with the

compressor during training would enable a fully end-to-end framework, potentially leading to better integration and overall performance.

REFERENCES

[1] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] L. Mochurad, "A comparison of machine learning-based and conventional technologies for video compression," *Technologies*, vol. 12, no. 4, 2024. [Online]. Available: https://www.mdpi.com/2227-7080/12/4/52

[4] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and Video Compression with Neural Networks: A Review," Apr. 2019, arXiv:1904.03567 version: 2. [Online]. Available: http://arxiv.org/abs/1904.03567

[5] J. Li, B. Li, and Y. Lu, "DCVC-FM: Neural Video Compression with Feature Modulation," Feb. 2024, arXiv:2402.17414 version: 1. [Online]. Available: http://arxiv.org/abs/2402.17414

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] Ultralytics, "YOLOv8 Documentation," https://docs.ultralytics.com/models/yolov8/, 2023, accessed: 2025-01-14.

[8] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," May 2018, arXiv:1802.01436 [eess]. [Online]. Available: http://arxiv.org/abs/1802.01436

[9] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end Optimized Image Compression," Mar. 2017, arXiv:1611.01704 [cs]. [Online]. Available: http://arxiv.org/abs/1611.01704

[10] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," 2020. [Online]. Available: https://arxiv.org/abs/2007.03034

[11] C.-H. Huang and J.-L. Wu, "Unveiling the future of human and machine coding: A survey of end-to-end learned image compression," *Entropy*, vol. 26, no. 5, 2024. [Online]. Available: https://www.mdpi.com/1099-4300/26/5/357

[12] J. Ascenso, E. Alshina, and T. Ebrahimi, "The JPEG AI Standard: Providing Efficient Human and Machine Visual Data Consumption," *IEEE MultiMedia*, vol. 30, no. 1, pp. 100–111, Jan. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10123093

[13] J. Li, B. Li, and Y. Lu, "DCVC: Deep Contextual Video Compression," Dec. 2021, arXiv:2109.15047. [Online]. Available: http://arxiv.org/abs/2109.15047

[14] ——, "DCVC-HEM: Hybrid Spatial-Temporal Entropy Modelling for Neural Video Compression," Jul. 2022, arXiv:2207.05894. [Online]. Available: http://arxiv.org/abs/2207.05894

[15] X. Sheng, J. Li, B. Li, L. Li, D. Liu, and Y. Lu, "DCVC-TCM: Temporal Context Mining for Learned Video Compression," Jan. 2023, arXiv:2111.13850. [Online]. Available: http://arxiv.org/abs/2111.13850

[16] J. Li, B. Li, and Y. Lu, "DCVC-DC: Neural Video Compression with Diverse Contexts," Mar. 2023, arXiv:2302.14402. [Online]. Available: http://arxiv.org/abs/2302.14402

[17] Z. Jia, B. Li, J. Li, W. Xie, L. Qi, H. Li, and Y. Lu, "Towards Practical Real-Time Neural Video Compression," Mar. 2025, arXiv:2502.20762 [eess]. [Online]. Available: http://arxiv.org/abs/2502.20762

[18] A. Mercat, M. Viitanen, and J. Vanne, "Uvg dataset: 50/120fps 4k sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 297–302. [Online]. Available: https://doi.org/10.1145/3339825.3394937

[19] Y. Li, S. Liu, Y. Chen, Y. Zheng, S. Chen, B. Zhu, and J. Lou, "An optimized h.266/vvc software decoder on mobile platform," 2021. [Online]. Available: https://arxiv.org/abs/2103.03612

[20] Y. Wang, P. H. Chan, and V. Donzella, "Semantic-aware video compression for automotive cameras," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 6, pp. 3712–3722, 2023.

[21] C. Cai, L. Chen, X. Zhang, and Z. Gao, "End-to-end optimized roi image compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 3442–3457, 2020.

[22] Y. Jiang, H. Zhang, L. Li, D. Liu, and Z. Li, "Sparse Point Clouds Assisted Learned Image Compression," Dec. 2024, arXiv:2412.15752 [cs]. [Online]. Available: http://arxiv.org/abs/2412.15752

[23] S. Dani, "Kitti object detection using yolov8n," https://www.kaggle.com/code/shreydan/kitti-object-detection-yolov8n/, 2023, accessed: 2025-01-14.

[24] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2018. [Online]. Available: https://arxiv.org/abs/1708.07120

[25] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," 2020. [Online]. Available: https://arxiv.org/abs/2011.03029

[26] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," Feb. 2015, arXiv:1405.0312 [cs]. [Online]. Available: http://arxiv.org/abs/1405.0312