Machine Learning-based Multimodal Data Compression

JACOB FORSELL & YUYANG JIN

MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Machine Learning-based Multimodal Data Compression

Jacob Forsell & Yuyang Jin

Supervised by Saeed Bastani, Alexander Ekman, Marcus Valtonen Örnhag, & Amir Aminifar

Thesis submitted for the degree Master of Science (30 hp)

Host company: Ericsson

Lund's Faculty of Engineering

Department of Electrical and Information Technology

January 2024 - June 2024

Abstract

The field of learned image compression has been facing rapid development and research engagement. In this thesis, we seek to make an addition to the field by extending a state-of-the-art learned image compression architecture, called LIC-TCM, by incorporating a depth map as a second complementary modality to further enhance the image compression. Furthermore, we explore the inverse, meaning we primarily compress a depth map (which can be expressed as an image) using LIC-TCM and incorporate the corresponding image frame as a secondary complementary modality. An important goal in this work has been to target low complexity in terms of encoding and decoding time, and implementation, something which is reflected in our proposed architectures.

In this thesis, we propose three unique architectures. The first architecture, Attention-based Multimodal-LIC-TCM, uses a depth map as a secondary complementary modality and expands the encoder such that it incorporates depth as a token into its SWIN transformer architecture. The second architecture, Convolution-based Multimodal-LIC-TCM, uses an image as a secondary complementary modality and uses a unique convolution-based module to extract features from both the image and depth modalities and subsequently fuses them for further enhanced compression. The third architecture, 4-channel LIC-TCM architecture, jointly accepts an image and depth as input, and either targets an image or depth map as reconstruction depending on configuration. However, for this last architecture, the LIC-TCM architecture itself has not been altered thus minimizing complexity.

Generally, we find that the three architectures show improved reconstruction performance while simultaneously not affecting the compressed size, but the improvement is not significant. However, the attention- and convolutional-based architectures introduce more time- and implementation complexity, while 4-channel LIC-TCM is not affected in this sense. These results indicate positive research directions, but further investigations are required for more conclusive results.

Acknowledgments

First of all, we would like to thank Ericsson for providing the resources and opportunity to conduct a master's thesis. A special thank you goes to Device Platform Research at Ericsson for facilitating such an inclusive and educative environment.

We would like to express our deepest gratitude to our main supervisor, Saeed Bastani (Ericsson), for his guidance and knowledge and for letting us explore this unique topic together with Ericsson. Your help has been essential and we could not have pulled this through without your outstanding help.

We are also very thankful for our co-supervisors Alexander Ekman (LU), Marcus Valtonen Örnhag (Ericsson) and Amir Aminifar (LTH). Our discussions together has been fruitful and important in our endeavours. A special thank you to Alexander, for going over and beyond with providing expertise and technical support in difficult times.

Lastly, we would like to thank our families and friends, whose encouragement and support have been invaluable in helping us achieve our goals and complete this thesis.

Contents

1 Introduction

“Data deluge” [1] refers to the situation where the sheer volume of new data being generated is overwhelming the storage and processing capacities of any single user device or server. This is becoming a common problem in both academia and industry. With the rapid advancement of information technology, various sectors including industries, academia, and individuals are generating substantial volumes of data daily, ranging from personal text messages to extensive datasets utilized in scientific endeavors. According to the International Data Corporation (IDC), the global digital data footprint reached 64.2 zettabytes (ZB) at the onset of 2020, marking a staggering increase of approximately 32 times compared to 2010. IDC predicts that this exponential growth will continue, with global data projected to expand to 181 zettabytes by 2025, see Figure 1.

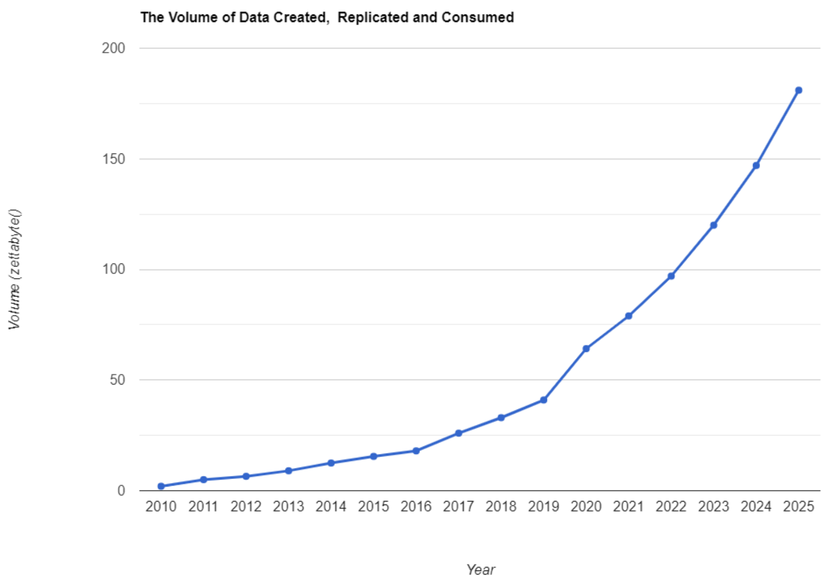


Figure 1: The Volume of Data Created, Replicated, and Consumed [2].

This surge in data volume naturally presents numerous challenges in terms of storage, processing, and transformation. Data, by its nature, necessitates storage space. As data quantities escalate, so does the space required for storage. While conventional practice involves storing data on cloud servers, this often entails significant expenses. Moreover, many scenarios demand real-time data processing, such as system monitoring, placing substantial strain on network performance. In networks with restricted bandwidth or high congestion, transmitting large data volumes at high speeds can precipitate bottlenecks and performance degradation.

The need for data compression is particularly critical in mobile devices, including XR (Extended Reality) devices. Modern mobile devices are equipped with a plethora of sensors, such as cameras, accelerometers, gyroscopes, GPS, microphones, and various biometric sensors. These sensors generate diverse data modalities including video, audio, location data, motion data, and biometric data.

For instance, an XR device might use its camera and depth sensors to capture and process high-resolution video and spatial data in real-time to create immersive augmented reality experiences. Simultaneously, the device might track user movements and gestures using accelerometers and gyroscopes while capturing audio inputs for interactive features. The volume of data generated from these sensors is substantial, necessitating efficient data management strategies.

The availability of multiple sensor modalities in a mobile device not only generates large volumes of data but also presents opportunities for joint processing and compression of this data. For example, by jointly processing video and audio data captured during an XR session, it would be interesting if it is possible to use contextual information from one modality to improve the compression efficiency of the other.

To address this challenge, researchers have developed various algorithms for data compression. Initially, the approach relied on traditional mathematical theories for compression. However, with the advancement of AI technology, neural networks have become extensively utilized in this domain, quickly surpassing traditional methods in some instances. Presently, machine learning-based approaches have emerged as a popular method for data compression. As a novel addition to this field, an investigation is carried out to explore if multiple modalities can be used with machine learning-based compression models to further enhance data compression.

In this thesis, we aim to address whether the two modalities, RGB and depth, can be used jointly to achieve superior image compression. If so:

- Can multimodal architectures achieve a lower bitrate while maintaining the same reconstruction quality? How do these models compare with each other?
- How much complexity do these multimodal models add, in terms of runtime and implementation?

In our work, we draw significant inspiration from two previous studies that investigated incorporating RGB and depth as modalities into their learned image compression models [3], [4]. However, these studies have limited citations, and the models they extended to include multimodal properties are slightly outdated. Therefore, we apply their approaches to a more state-of-the-art model [5] to gain further insights. Furthermore, for one of the architectures (later denoted as 4C LIC-TCM), we adopt a very simple design process to create our own custom architecture which is not directly inspired by some paper.

To summarize, we propose three different unique multimodal architectures that jointly accept both RGB and depth as input and compress with one of these modalities targeted as output. The architectures are based on different multimodal strategies intended for machine learning and are built separately, trained separately, and evaluated separately.

In Sections 2.1, 2.2, and 2.3, we will cover the necessary background to understand the concepts and ideas that reside within this thesis. Section 2.4 covers the background behind the foundational models that will be further extended to incorporate multimodal properties. These extended architectures, dataset, test dataset, experimental setup, and more are covered in Section 3. Thereafter, the results are covered in Section 4 and discussed in Section 5. Finally, some conclusions and outlook are provided in Section 6, which will conclude this work.

2 Background and Related work

In this section, we will delve into the foundational concepts and relevant research that underpin the field of multimodal data compression. In Section 2.1, we begin exploring "Artificial Neural Networks and Deep Learning", highlighting the advancements and techniques that have significantly enhanced data compression capabilities. Next, in Section 2.2, we cover "Information Theory", which provides the theoretical underpinnings for data compression and understanding Rate-Distortion optimization objectives. Following this, in Section 2.3, we examine "Multimodal Machine Learning", focusing on the integration and processing of diverse data types to achieve more robust and efficient models. Finally, in Section 2.4, we cover "Foundational Models", which include the essential background of the pivotal architectures that form the basis of our work. In other words, this final subsection will discuss related works on learned image compression and previous work on potential use of multiple modalities to further enhance compression.

2.1 Artificial Neural Networks and Deep Learning

The architecture of Artificial Neural Networks (ANNs) draws inspiration from the structure and functionality of the human brain. Typically, an artificial neural network comprises three layers: the input layer, the hidden layer, and the output layer. Each layer consists of interconnected nodes, also known as artificial neurons. During operation, these artificial neurons (see Figure 2) receive inputs from other nodes, sum them up, pass the result through an activation function, and ultimately transmit the output to other connected nodes. The process can be defined as

$$y = \phi(\sum_{k=1}^{K} \omega_k x_k + b) \tag{1}$$

where ω_k, x_i, b, y represent the weights, inputs, the bias, and the output of the node respectively and $\phi(\cdot)$ is the activation function. It is important to note that communication between nodes is restricted to those that are directly connected [6].

Deep learning, a subset of machine learning algorithms, treats each node as a feature of the input object. Taking the classification of an image of an animal as an example, a simple neural network with three hidden layers might involve edge detection in the first layer, where each node detects different types of edges in the input image [8]. The subsequent layers could progress to detecting more complex features, such as identifying eyes based on the edges detected in the previous layer, and the final layer might engage in more generalized tasks, like recognizing whether the animal has wings.

The applications of deep learning are extensive in modern society, encompassing automatic speech recognition, natural language processing (NLP), and medical image analysis. As artificial intelligence advances, online disease diagnosis becomes a viable prospect. In essence,

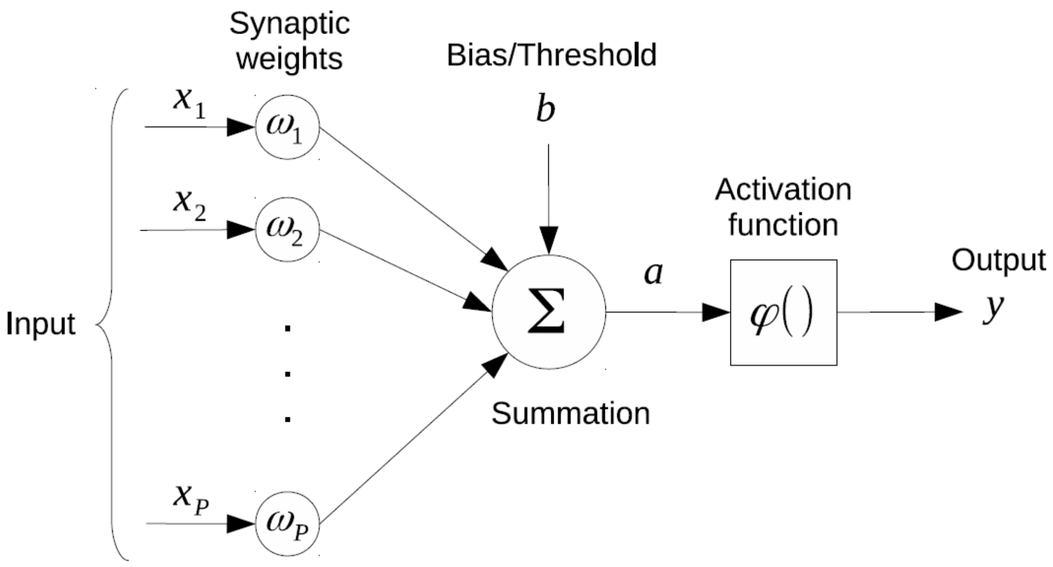


Figure 2: The basic element of the ANN, the neuron [7].

deep learning serves as a driving force for innovation, transforming various industries by providing solutions to intricate problems and unlocking opportunities for automation, efficiency, and enhanced decision-making.

2.1.1 Dense Neural Networks

A dense neural network is also known as a fully-connected neural network. In a dense neural network, each node is connected to all the nodes in the subsequent layer. This architecture contrasts with other types of neural networks like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), which have specialized layer structures. Figure 3 presents the typical architecture of a dense neural network.

2.1.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) represents a prevalent form of artificial neural network, particularly utilized in tasks related to image and video recognition [10]. The fundamental components of a CNN include convolutional layers, pooling layers, and fully-connected layers. Within convolutional layers, the network applies various filters to the input image, enabling feature extraction and hierarchical representation learning [10]. The operation of convolution can be represented as

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]. \tag{2}$$

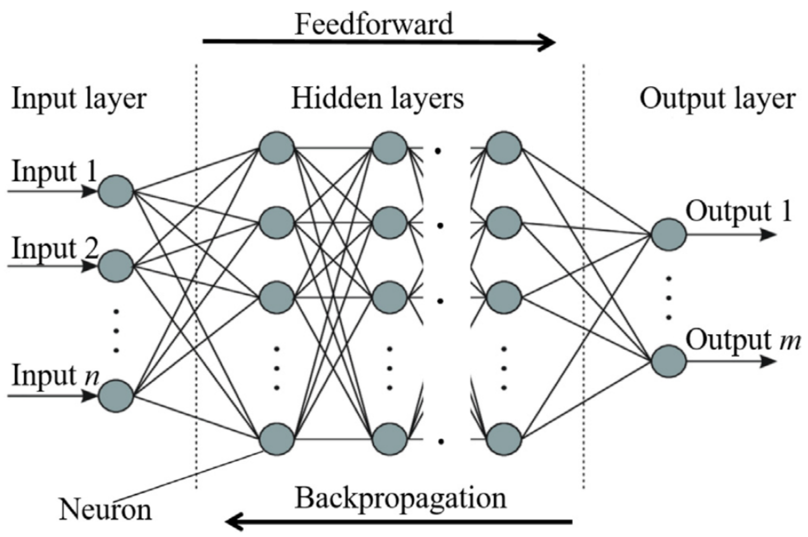


Figure 3: An illustration of the dense neural network [9].

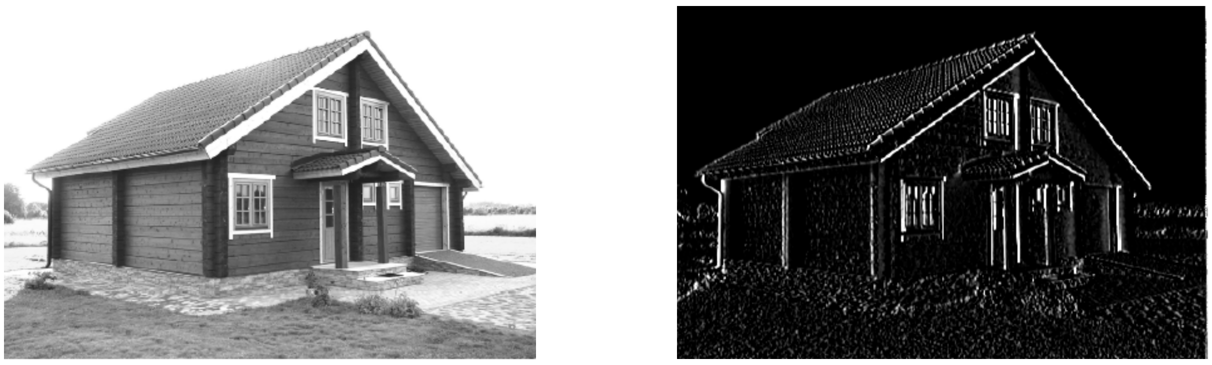
The objective of the convolutional layer is to extract features from the input image. Figure 4 illustrates the implementation of a filter designed to detect vertical edges within the input image. Following the convolutional layer is the pooling layer, which employs strategies such as max-pooling or average-pooling. In max-pooling, for instance, the largest value within a specified area is used to represent that region. An example of max-pooling is depicted below

$$\begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \\ 9 & 10 & 13 & 14 \\ 11 & 12 & 15 & 16 \end{bmatrix} \xrightarrow{max-pooling} \begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix}. \tag{3}$$

The primary function of the pooling layer is to decrease dimensionality, thereby reducing the computational complexity of the network. In a typical convolutional neural network, multiple convolutional and pooling layers are commonly employed. Once an adequate number of features are extracted, the outputs are flattened into a long one-dimensional vector, serving as the input for a dense neural network.

2.1.3 Autoencoders

An autoencoder (AE) is a type of artificial neural network that has three major components: an encoder, a latent space, and a decoder. In general, the objective of an autoencoder is to learn efficient data representation (or coding), typically for dimensionality reduction [11]. Thus, it can be used for compression purposes or even generative purposes by sampling the latent space. To achieve this, the encoder part of the network is trained to efficiently map input to the latent space and the decoder maps this representation back to the same space as the input. An overview of the autoencoder network can be seen in Figure 5.



(a) The original input image. **(b)** The image with a vertical edge detector.

Figure 4: The images before and after applying a filter [7].

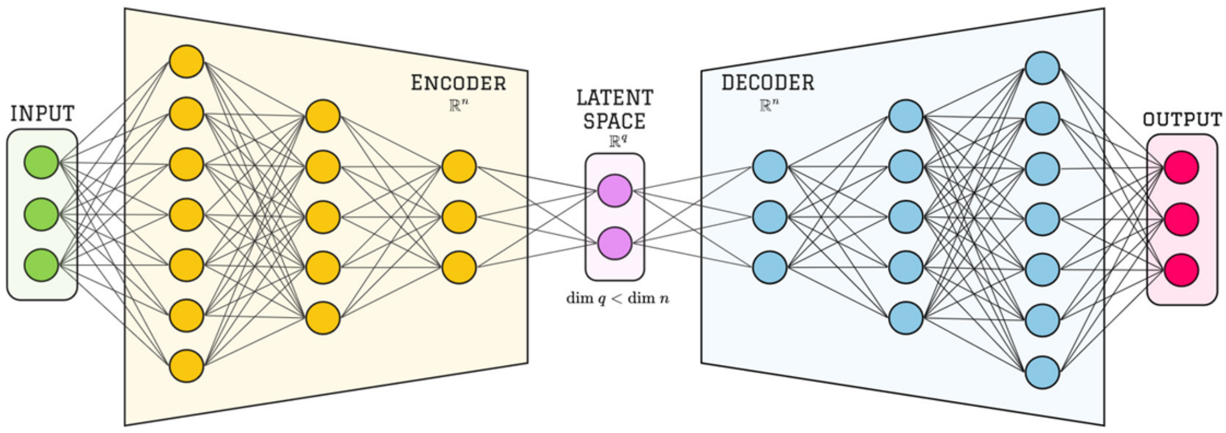


Figure 5: An illustration of an autoencoder network.

Most basic autoencoders can be described as having an encoder function, $E_\phi : \mathbb{R}^n \longrightarrow \mathbb{R}^q$, that depends on some parameters set by ϕ and evaluated on some input dataset $\mathcal{X} = \{x^{(i)}\}_{i=1}^N$ consisting of N i.i.d samples. The latent feature representation can then be described as

$$z_i = E_\phi(x_i), \tag{4}$$

where each latent variable $y \in \mathcal{Y}$ and $\mathcal{Y} \in \mathbb{R}^q$ defines the latent space. Note that $q < n$ hence dimensionality reduction. Conversely, the decoder function $\mathcal{D}_\theta : \mathbb{R}^q \longrightarrow \mathbb{R}^n$, depends on some parameters set by θ and evaluates on any $y \in \mathcal{Y}$. The output of the decoder function, the reconstruction message, is given by

$$\hat{x}_i = \mathcal{D}_\theta(z_i) = \mathcal{D}_\theta(E_\phi(x_i)), \tag{5}$$

where $\hat{x}_i \in \mathbb{R}^n$. Training an autoencoder simply means that, as given by [12], finding the functions $E_\phi(\cdot)$ and $\mathcal{D}_\theta(\cdot)$ that satisfy

$$\underset{E_\phi, \mathcal{D}_\theta}{\operatorname{argmin}} \langle \Delta(x_i, \mathcal{D}_\theta(E_\phi(x_i))) \rangle, \tag{6}$$

where Δ indicates a measure of how the input and the output of the autoencoder (penalizing the difference between input and reconstructed output) and $\langle\cdot\rangle$ indicates the average over all observations.

Most autoencoder-based compression tools follow this workflow in one way or another, but the architecture behind an autoencoder can vary greatly (as the reader will see soon). The standard autoencoder architecture is too limiting in many cases, especially for compression and generative purposes. The issue often stems from the fact that a standard autoencoder is too "rigid" and has a simple discrete mapping process. In other words, the standard autoencoder does not feature any good regularizations to the latent space such that it incorporates properties such as continuity and completeness. However, one type of architecture that supports this and is particularly interesting in relation to lossy image compression is Variational Autoencoders (VAE).

2.1.4 Transformers

Transformers were initially introduced in [13], where this newly proposed architecture outperforms the other state-of-the-art models in machine translation tasks. Compared with Convolutional Neural Networks (CNN), transformers are proven to have the advantage of taking both local and global information into consideration [5]. It also follows the encoder-decoder structure. In both the encoder and the decoder part, the transformer uses a novel "attention mechanism" to compute the latent relation between inputs and outputs, instead of relying on the traditional "recursive structure" [13]. By using the attention mechanism, the transformer can comprehend context and meaning by analyzing the relationship in the elements of its input. However, in general, to achieve well-learned understanding of the inputs' context and meaning, the transformers require a lot of data to be trained on [14].

2.1.4.1 Vision Transformers As mentioned before, transformers are initially designed for machine translation tasks. It is thus natural to explore the possibility of its application in visual tasks. Dosovitskiy et al. proposed a *vision transformer (ViT)* [15] which directly applies the transformer to images. Similar to CNNs, vision transformers can also capture spatial relationships between different parts of an image, but they rely on the self-attention mechanisms instead.

In essence, vision transformers break down an image into smaller patches and process them individually through a series of transformer layers. These transformer layers allow the model to learn complex relationships between these patches, enabling it to understand the global context of the image [15].

One of the key advantages of vision transformers is their ability to handle both local and global information effectively, without relying on handcrafted features or hierarchical feature extraction like in CNNs. This makes them highly adaptable to different types of visual tasks and datasets [15].

2.1.4.2 SWIN Transformers SWIN transformers address some of the limitations of Vision Transformers, such as their reliance on fixed-size patches and lack of efficient hierarchical processing [16]. The innovations mainly manifest in two aspects: Hierarchical Processing and shifted windows. Firstly, SWIN transformers process images in a hierarchical manner, dividing them into smaller windows and processing them at multiple scales. This hierarchical approach allows it to capture both local and global information effectively, enhancing its ability to understand images. Moreover, SWIN Transformer employs shifted windows to enable overlapping between adjacent patches. This ensures that each pixel is covered by multiple windows during processing, enhancing the model's ability to capture fine-grained details and spatial relationships.

2.1.5 Variational Autoencoders

The variational autoencoder is a variational Bayesian-based architecture, which shares architectural affinities with a standard autoencoder but stands on a completely different mathematical foundation and objective. This specially designed autoencoder makes it possible to jointly learn *deep latent-variable models* and corresponding inference models using stochastic gradient descent [17], [18].

In essence, variational autoencoders are probabilistic generative models. The general idea is that the encoder is parametrized by some probability distribution which learns stochastic mappings between an input space to the latent space. The stochastic decoder inverses this process, and samples the latent space based on the decoder's parametrized probability distribution. This architecture has shown to be promising in multiple fields, such as compression and generative (image, video, etc.) purposes [19], [20]. While the above has been a greatly simplified description of VAEs, what will follow is a more detailed walkthrough that heavily relies on the previously mentioned introductory paper [18]. Note that for this section we are not precisely following the notations as given in 2.1.3. Furthermore, bold variables represent a set of underlying variables such that it is represented as a single vector.

To put things in more mathematical terms, assume an observed variable $\boldsymbol{x}$ which is a random sample from an *unknown underlying process*, whose true probability distribution is unknown. This can be interpreted as a generative model, where the decoder network is sampling a latent variable $\boldsymbol{y}$ from some prior distribution, commonly a normal distribution, in the latent space.

Now, assume there is a deterministic function, $f_\theta(\boldsymbol{y})$, which maps $\boldsymbol{y}$ to the parameters ψ, where f represents a neural network parameterized by θ. Then, by defining some parametric distribution $\mathcal{D}$ (commonly a normal distribution), one can sample from the latent space and generate some samples $\boldsymbol{x}$, where $\boldsymbol{x} \sim \mathcal{D}(\psi)$. Although the prior- and $\mathcal{D}$-distribution are quite simple, the generative model can fit very complicated distributions because of its non-linear mapping between $\boldsymbol{y}$ and ψ via the neural network.

Kingma and Welling [18] use the term *deep latent-variable model* to denote a latent variable model $p_\theta(\boldsymbol{x}, \boldsymbol{y})$ (joint distribution over both observed variable $\boldsymbol{x}$ and latent variables $\boldsymbol{y}$) whose distributions are parameterized by neural networks. One can then express the marginal likelihood $p_\theta(\boldsymbol{x})$, given the parameters θ, by marginalizing over $\boldsymbol{y}$ and get

$$p_\theta(\boldsymbol{x}) = \int_{\boldsymbol{y}} p_\theta(\boldsymbol{x}, \boldsymbol{y}) dz = \int_{\boldsymbol{y}} p_\theta(\boldsymbol{x}|\boldsymbol{y}) p(\boldsymbol{y}) dz. \tag{7}$$

If one considers the joint distribution

$$p_\theta(\boldsymbol{x}, \boldsymbol{y}) = p_\theta(\boldsymbol{x}|\boldsymbol{y}) p(\boldsymbol{y}), \tag{8}$$

one can observe two important distributions, the prior distribution of the latent space $p(\boldsymbol{y})$, and the decoder's conditional likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{y})$. The latter distribution is directly related to the previously mentioned $\mathcal{D}$.

As soon will be explained, the marginal likelihood $p_\theta(\boldsymbol{x})$ is a key component in optimizing the variational autoencoder, but since it is intractable due to (7) not having an analytic solution or efficient estimator, approximate inference techniques are needed. Because of this, the posterior $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ is also intractable, clearly displayed by the Bayes' rule

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}, \boldsymbol{y})}{p_\theta(\boldsymbol{x})}. \tag{9}$$

To turn the problem from an intractable problem to a tractable problem, the authors introduce a parametric *inference model* $q_\phi(\boldsymbol{y}|\boldsymbol{x})$ where the variational parameters ϕ are optimized such that

$$q_\phi(\boldsymbol{y}|\boldsymbol{x}) \approx p_\theta(\boldsymbol{y}|\boldsymbol{x}). \tag{10}$$

This newly introduced posterior distribution represents the encoder network and creates a link to the structure of a vanilla autoencoder. To clarify, in simple terms, we now have an encoding network represented by $q_\phi(\boldsymbol{y}|\boldsymbol{x})$, which maps an input to the latent space. This latent space is assumed to follow a prior p. This latent space can then be sampled by the decoder network $p_\theta(\boldsymbol{x}|\boldsymbol{y})$. The reader should now be ready to comprehend the elements that are needed to state the optimization objective.

2.1.5.1 Optimization objective The optimization objective of the variational autoencoder is to maximize the evidence lower bound (ELBO). This objective encompasses two components, with regards to the parameters θ and ϕ (in simple terms)[18]:

- maximize the marginal likelihood $p_\theta(\boldsymbol{x})$, and
- minimize KL–divergence between $p(\boldsymbol{y})$ and $p_\theta(\boldsymbol{y}|\boldsymbol{x})$.

Given the inference model in (10), one can get (Eq. 2.8 in [18])

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{y}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x})] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{y}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{y})}{p_\theta(\boldsymbol{y}|\boldsymbol{x})}\right]\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{y}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{y})}{q_\phi(\boldsymbol{y}|\boldsymbol{x})}\frac{q_\phi(\boldsymbol{y}|\boldsymbol{x})}{p_\theta(\boldsymbol{y}|\boldsymbol{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{y}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{y})}{q_\phi(\boldsymbol{y}|\boldsymbol{x})}\right]\right]}_{=\mathcal{L}_{\theta,\phi}(\boldsymbol{x})} + \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{y}|\boldsymbol{x})}\left[\log\left[\frac{q_\phi(\boldsymbol{y}|\boldsymbol{x})}{p_\theta(\boldsymbol{y}|\boldsymbol{x})}\right]\right]}_{=\mathcal{D}_{KL}(q_\phi(\boldsymbol{y}|\boldsymbol{x})||p_\theta(\boldsymbol{y}|\boldsymbol{x}))}
\end{aligned} \tag{11}$$

where the first term $\mathcal{L}_{\theta,\phi}(\boldsymbol{x})$ and the second term $\mathcal{D}_{KL}(q_\phi(\boldsymbol{y}|\boldsymbol{x})||p_\theta(\boldsymbol{y}|\boldsymbol{x}))$ are the evidence lower bound (ELBO) and the Kullback–Leibler (KL) divergence, respectively. Note that the first equality in (11) may cause some confusion, but this is done by the authors to avoid deriving the ELBO through Jensen's inequality for pedagogical reasons. The reader is referred to [18] for further details. Rewriting (11) we get

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = \log p_\theta(\boldsymbol{x}) - \mathcal{D}_{KL}(q_\phi(\boldsymbol{y}|\boldsymbol{x})||p_\theta(\boldsymbol{y}|\boldsymbol{x})). \tag{12}$$

Once again, the optimization objective is to maximize ELBO $\mathcal{L}_{\theta,\phi}(\boldsymbol{x})$ with regards to the parameters θ and ϕ. Since KL divergence is non-negative, the log-likelihood of the data sets the lower bound of the ELBO. Based on the objective to maximize ELBO, two observations can be made (see Section 2.2.1 in [18]). Firstly, the marginal likelihood $p_\theta(\boldsymbol{x})$ should be maximized and secondly, the KL-divergence should be minimized such that the gap between the approximation $q_\phi(\boldsymbol{y}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ will decrease.

However, (11) does not tell the entire story. In order for the objective to be feasible for machine learning-based training, it must allow for *joint* optimization with regards to θ and ϕ using Stochastic Gradient Descent (SGD). The path forward of using SGD is not trivial and introduces complications and solutions such as *reparameterization tricks* are needed. While much more can be said (sampling techniques, etc.) on the topic of variational autoencoders, the reader can refer to [18] for more details. The reader will now be provided the final loss function for a VAE which is the negative ELBO

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = -(\underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{y})]}_{\text{Reconstruction Loss}} - \underbrace{\mathcal{D}_{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x})||p(\boldsymbol{y})]}_{\text{Regularization Term}}). \tag{13}$$

The reconstruction loss is self-explanatory, and the regularization term assures that the latent variables are distributed according to the prior distribution.

Finally, by assigning a hyperparameter λ to (13), one can determine the trade-off between the reconstruction loss and regularization term:

$$\mathcal{L}_{\theta,\phi}(x) = -(\lambda \cdot \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{y})]}_{\text{Reconstruction Loss}} - \underbrace{\mathcal{D}_{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x})||p(\boldsymbol{y})]}_{\text{Regularization Term}}).$$

2.1.6 Variational Image Compression

Since efficient compression often relies on knowledge of the probabilistic structure of the data, the problem is closely related to probabilistic source modeling, hence the application of variational autoencoders has been a natural path forward and thus has been extensively researched. More specifically, the area of lossy image compression has made great progress and is already showing results that supersede standard methods such as JPEG and even BPG [3]–[5].

The general idea is to refashion the general optimization objective of a variational autoencoder (see (13)) to a Rate-Distortion optimization problem. In general, a variational autoencoder (similar to a standard autoencoder) generates a lower dimensional representation in the latent space which is represented by some floating values. In the context of compression, a quantization operation can be used on the floating latent representation and thereafter be losslessly compressed. Since quantization converts the floating latent representation to an integer-valued (or discrete) representation, it can thereafter be losslessly compressed using some entropy coding such as rANS (see Section 2.2.1.1). However, since quantization introduces error, which is tolerated in the context of lossy compression, the optimization objective can now be interpreted as a Rate-Distortion optimization objective instead.

One important contribution to the field of lossy image compression is *Variational image compression with a scale hyperprior* by Ballé et al. [21]. The paper proposes an end-to-end trainable model for lossy image compression based on variational autoencoders where it extends previous work [22] and, most importantly, introduces a powerful entropy model by incorporating a hyperprior on the local scale parameters of the latent representation. In other words, it uses context-adaptive entropy models to achieve superior compression performance. Many recent papers (in relation to lossy image compression) are based on the development of the work behind learned entropy models and since our foundational architecture (see Section 2.4.1) uses one of the state-of-the-art models, this discussion will provide the reader with the necessary insights to understand the rationale behind the design of our future loss function (see Section 2.4.1.4).

Assume a autoencoder-based architecture, as explained in Section 2.1.5. Then, introduce a quantization function and entropy coding algorithm within the latent space. However, since entropy coding requires a prior probability model of the quantized representation, which is known to both the encoder and decoder, an entropy model of some kind has to be constructed. Therefore, a new separate variational autoencoder network representing the entropy model is attached to the latent space and accepts the latent variable $\boldsymbol{y}$ as input. In the same way an encoder (g_a) and decoder (g_s) network is used in the compression model, the entropy model has an encoder and decoder network represented by h_a and h_s, respectively. The overall goal of the entropy model then becomes to learn how it can most efficiently compress a bit-stream and simultaneously learn an efficient latent representation of the entropy model itself. See Figure 6 for an overview of the variational image compression model.

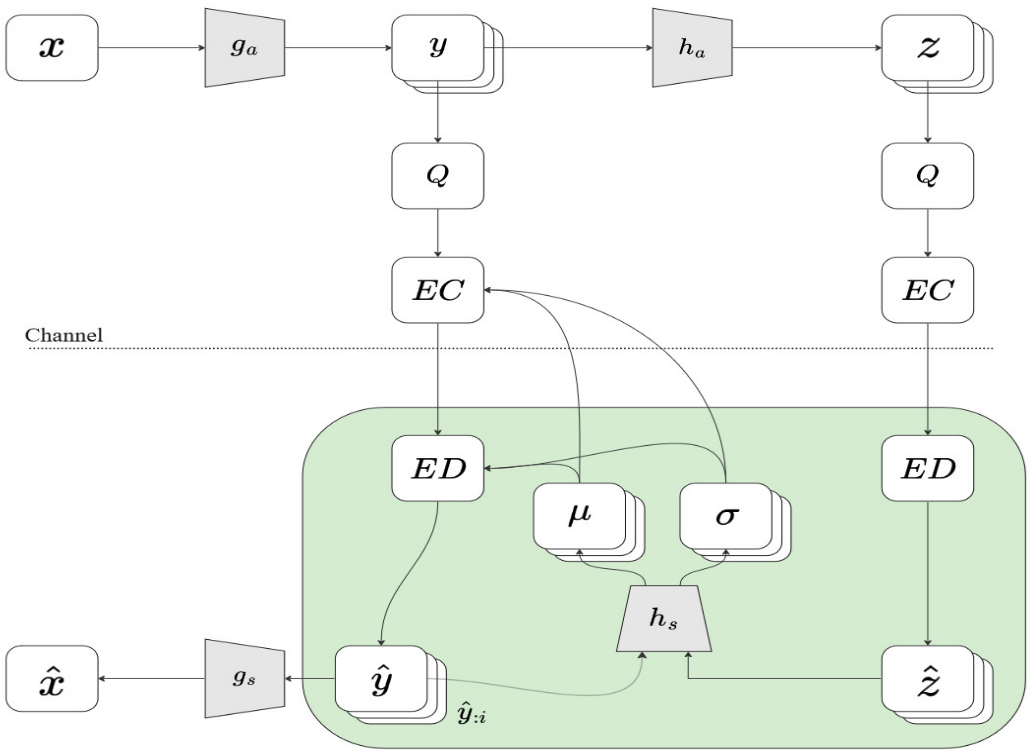


Figure 6: A simplified illustration of a variational image compression model, inspired by Figure 10 in [23]. An input $\boldsymbol{x}$ is fed to the compression model through an encoder network g_a and a decoder network g_s to generate the reconstruction $\hat{\boldsymbol{x}}$. The novelty of Ballé et al. is introduced in the inclusion of quantization operation (Q), the entropy coding (EC), the entropy decoding (ED) and the entropy model featuring an encoder network (h_a) and decoder network (h_s). In this communication system, the transmitter entity (or sender) has to both compute the upper diagram and the shaded box. The latter is necessary to compute because the entropy model needs to be used to compute the entropy coding. However, the receiver only has to calculate the shaded box if given the entropy coded latent representations.

Combining these two networks, the compression model and the entropy model, seems to introduce significant mathematical complexity. However, it can be shown that during right circumstances, this given model can in fact be overall interpreted as a variational autoencoder [24]. Therefore, the interested reader can refer to papers covering the topic in more detail [21]–[24]. Thus, omitting further architectural details, the loss function of the variational image compression by Ballé et al. can now be established (see Eq. 10 in [21]). Let $\boldsymbol{y}$ denote the latent variable from the compression model (output of previously mentioned g_a) and let $\boldsymbol{z}$ instead denote the latent variable from the entropy model (output of previously mentioned h_a). Then the loss function can be described as

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}\mathcal{D}_{\mathrm{KL}}[q||p_{\boldsymbol{y},\boldsymbol{z}|\boldsymbol{x}}] = \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}\mathbb{E}_{\boldsymbol{y},\boldsymbol{z}\sim q}(&\log q(\boldsymbol{y},\boldsymbol{z}|\boldsymbol{x}) - \log p_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{x}|\boldsymbol{y}) \\ &- \log p_{\boldsymbol{y}|\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{z}) - \log p(\boldsymbol{z})) + \text{const.}\end{aligned} \tag{14}$$

Note that the equation above has been simplified in notation for clarity. To expand further on (14), the equation can be further simplified according to [21]: The first term evaluates to zero, the second term is the distortion term (reconstruction loss) and the third term and the fourth term represent the bit rate in the compression model and entropy model, respectively. Hence, a rate-distortion objective is achieved but with a strong background in variational autoencoders.

Some further insights into the topic of rate-distortion can be read in Section 2.2.2.

2.2 Information Theory

Considering the fusion between variational Bayesian methods with autoencoders (see Section 2.1.6) which introduces some probabilistic interpretation, a connection between information-theoretic properties is introduced. Furthermore, considering that the field of data compression itself embodies concepts such as entropy coding, it is important that the reader receives some clarity on the topic. In this section, the essential related topics for this thesis within information theory will be covered. Firstly, the concept of data compression will be introduced, explaining the basic ideas behind lossy and lossless compression. Thereafter, the concept of entropy coding will be explained. Finally, a key concept, Rate-Distortion Theory will be introduced and which is directly related to lossy compression.

2.2.1 Compression

Data compression is a fundamental topic in science and engineering domains and is commonly formulated with the goal of designing codes for a given discrete data ensemble with minimal entropy [22], [25]. In other words, compression overall is the process of encoding information and reducing the original representation to a lower-dimensional representation. In terms of data compression, the practical process of encoding is done by reducing the amount of bits. This lower-dimensional representation can then in return be decompressed to its original representation through a decoder. Any compression is either lossy or lossless, meaning some information is lost in the process of compressing or no information is lost at all. The trade-off in lossy compression is usually between the integrity of the original data versus greater data reduction. Lossless compression on the other hand reduces bits by reducing any statistical redundancy, but may not reach the same compression strength as lossy compression. An autoencoder (see Section 2.1.3) is inherently lossy and therefore the focus of this thesis is set on lossy compression rather than lossless compression.

2.2.1.1 Entropy coding Entropy coding is simply the process of compressing. More specifically, entropy coding is the process of attempting to exploit the statistical redundancy of a source symbol sequence (in our case, bits) and to approach the lower bound of Shannon's source coding theorem [25]. Up until ten years ago or so, two approaches were considered the main approaches for entropy coding: Huffman coding and arithmetic/range coding [26]. They have a clear trade-off, where Huffman coding is fast but delivers lower compression rates and arithmetic/range coding approaches theoretical rate limit, but at a cost of heavy computational cost. However, in 2013, Jarek Duda published a new proposed entropy coding method, Asymmetric Numeral Systems (ANS) in [26]. The contribution and impact of this paper have clearly been large for the field of entropy coding, and is today used in a wide range of different compression applications.

In this work, a variant of ANS is used, called Ranged Asymmetric Numeral Systems (rANS). The details of this entropy coding are omitted, but since it plays an important role in the entropy model (see Section 2.4.1) it is important that the reader is aware of this. The variant rANS has been used in several machine learning-based lossy image compression architectures that use entropy models. In this thesis, rANS was already integrated to the LIC-TCM architecture (see Section 2.4.1) and we did not see a good reason to change it to some other entropy coding scheme.

2.2.2 Rate-Distortion

For lossy learned image compression, rate-distortion is a paramount metric that is used for evaluating the loss function. Rate-distortion objectives can be applied in many different scenarios, but for the sake of simplicity it will be strictly explained in relation to lossy image compression. In its most simple form, an example loss function for the field of lossy learned image compression can be stated as

$$\mathcal{L} = \mathcal{R} + \mathcal{D}, \tag{15}$$

for some given rate term $\mathcal{R}$ and distortion term $\mathcal{D}$. In simple terms, the rate term addresses the issue of minimizing the representation of a source symbol sequence, in this case bits, and the distortion term calculates the difference between the quality of the source versus the reconstruction based on some appropriate algorithm.

For lossy learned image compression, there are usually three important properties included. Firstly, the rate term $\mathcal{R}$ is usually calculated in terms of bits-per-pixel (bpp). In other words, bpp calculates on average how many bits are used per pixel in the image. Secondly, the distortion term $\mathcal{D}$ is often decided between Mean Square Error (MSE) or Multi-scale Structural Similarity Index Measure (MS-SSIM, see Section 3.4.3). Since bpp and MS-SSIM are very important but unusual metrics, further details regarding these metrics can be found in Section 3.4. Thirdly, a hyperparameter λ is often introduced to provide a rate-distortion trade-off. Larger λ means higher precision but worse compression (meaning higher compression sizes). The loss function can now be updated to

$$\mathcal{L} = \mathcal{R} + \lambda \cdot \mathcal{D}. \tag{16}$$

The configuration of λ depends on the choice of distortion algorithm. Example of different values can be found in Section 2.4.1.4.

2.3 Multimodal Machine Learning

The field of multimodality within machine learning is one of multi-disciplinary research. Modal refers to a way of expressing or perceiving things, and each source or form of information can be called a modal [27]. There is no strict definition of modality, but in general terms, it refers to "a way in which something happens or is experienced" [28]. Objects often present themselves through various modalities. For example, a dog can be characterized by an image, audio, textual description, etc. When we explore the relationships between various modalities or integrate different modalities for tasks such as recognition or prediction, we delve into the field of multimodality. In artificial intelligence, multimodality plays a crucial role in enabling computers to comprehend objects or behaviors, mirroring the capabilities of the human brain.

2.3.1 Categories and Representations

The digitalization of various modalities is the initial step for analysis, and it is often challenging because different modalities may require distinct representation spaces. For instance, image modalities are commonly represented by pixels, while text is typically symbolic [28]. Below, some typical representations of modalities will be provided to illustrate their digitalization.

An RGB image is represented as a combination of three color channels: red, green, and blue. Each pixel in the image is typically represented as a combination of intensity values for these three colors. In Figure 7, each color channel can have intensity values ranging from 0 to 1. Therefore, a pixel in such an image is represented by three numbers: one for the intensity of red, one for green, and one for blue. This allows for the creation of a wide range of colors by varying the intensities of these three primary colors. The combination of these three channels forms the full-color image, where each pixel's color is a mixture of red, green, and blue light. The RGB color model is the most common way to represent and display images on electronic systems like computer monitors, televisions, and digital cameras.

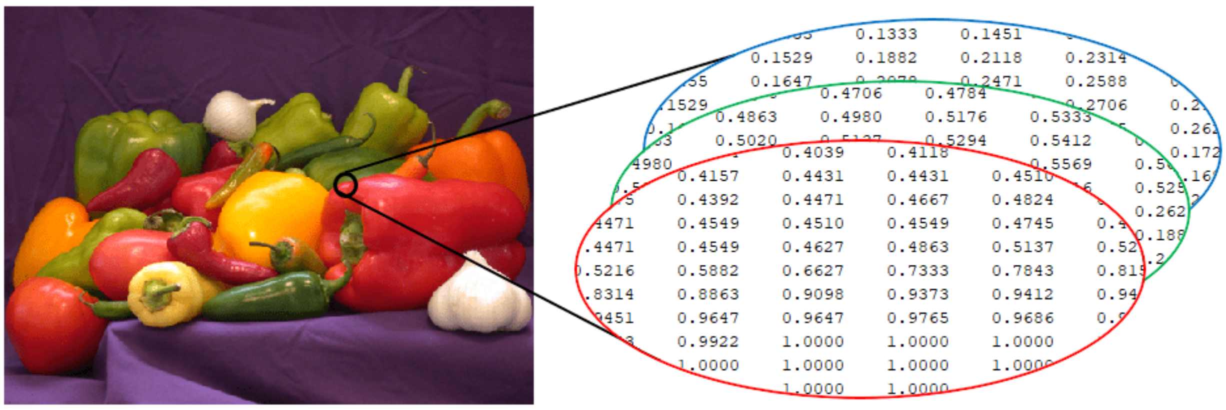


Figure 7: An example of the typical representation of a RGB image.

Likewise, a depth image is typically represented as a 2D array where each pixel value corresponds to the distance from the camera to the corresponding point in the scene. This distance value is often expressed in units such as meters or millimeters, depending on the calibration of the camera. They can be stored in various formats such as grayscale images, where darker pixels represent objects closer to the camera and lighter pixels represent objects farther away, see Figure 8.

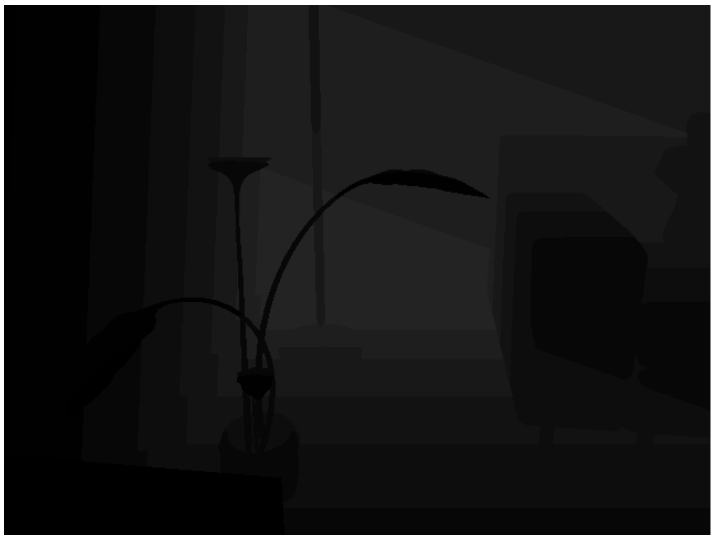


Figure 8: An example of the typical representation of a depth image. Note that the brightness and the contrast of the image are enhanced for better display.

2.3.2 Multimodal Fusion

Multimodal fusion refers to the integration of information from multiple modalities, such as text, images, audio, and video, to enhance understanding or perform a task. It involves combining the features or representations extracted from each modality into a unified framework, often using machine learning techniques. The goal is to leverage the complementary strengths of different modalities to achieve improved performance in tasks like classification, recognition, or generation.

As introduced in [28], the benefits of multimodal fusion can be listed in three points. Firstly, accessing multiple modalities observing the same phenomenon can enhance prediction robustness, notably utilized in the AVSR (audio-visual speech recognition) field. Take the application of multimodality in a classification scenario as an example, distinguishing between a dog and a wolf can be challenging if the input image is blurry or taken from a distance. However, when additional modalities are available, such as the sound of barking, the task can become considerably easier. Secondly, utilizing multiple modalities enables the capture of complementary information not discernible in individual modalities alone. Lastly, a multimodal system remains functional even if one modality is absent, such as recognizing emotions from visual cues when audio input is unavailable. In the experiment of Pavlov's dog, the sound of the bell alone was enough to trigger salivation, without the presence of food itself. This also reveals that human and most animals' brains also work in a multimodal fashion.

2.3.2.1 Fusion approaches Multimodal fusion can be divided into two categories, namely model-agnostic fusion and model-based fusion. The former is simpler and the fusion itself does not require any training, while the latter usually requires training of the neural network and is generally more complicated [28].

Multimodal fusion historically relies on model-agnostic methods, categorized into early, late, and hybrid fusion. Early fusion immediately integrates features upon extraction, often by combining their representations directly. It aims to capture correlations and interactions between low-level features of each modality. Conversely, late fusion happens when the analyses of the two modalities have already been made. It utilizes decision values from each modality and employs various fusion mechanisms like averaging or voting schemes. Lastly, hybrid fusion is a combination of early fusion and late fusion. It is considered a good way to maintain both low-level feature interaction and flexibility at the same time [28]. The following Figures 9 10 11 illustrate the workflow of the three fusion strategies.

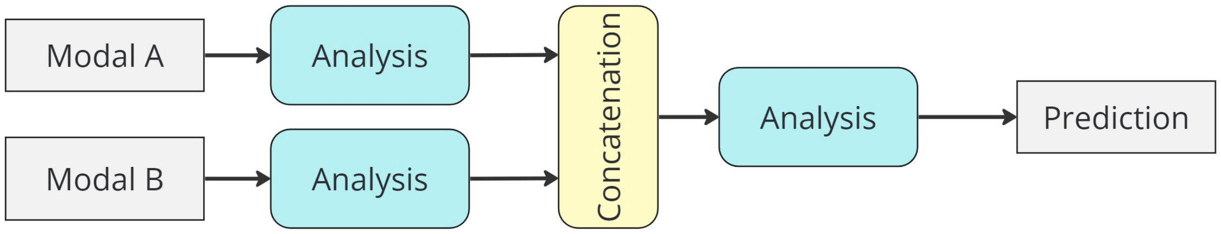


Figure 9: Early fusion.

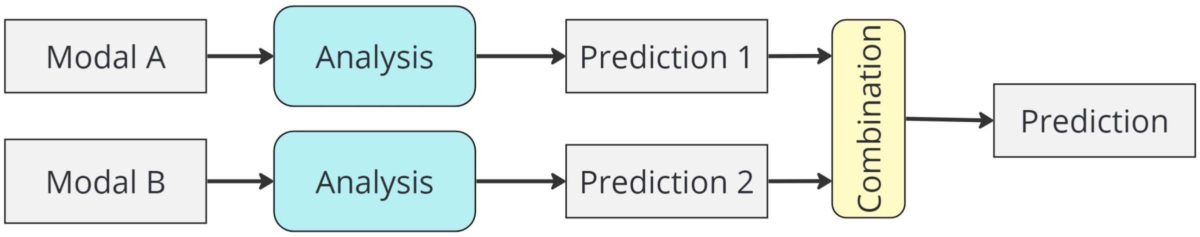


Figure 10: Late fusion.

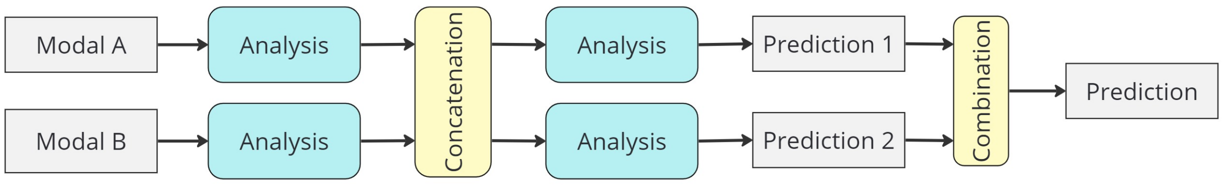


Figure 11: Hybrid fusion.

Model-based fusion depends on machine learning algorithms to build connections between two modalities [28]. Neural network-based fusion has now become the most popular way among all. Deep neural network approaches offer several advantages in data fusion. They excel at learning from vast amounts of data, leveraging their capacity to capture complex patterns across multiple modalities. Modern neural architectures enable end-to-end training, seamlessly integrating both the multimodal representation and fusion components into a unified framework. Figure 17 in section 3.1.2 gives a typical example of model-based fusion.

2.4 Foundational Models

As mentioned in the introduction (see Section 1), our objective is to identify an efficient method for compressing multimodal data. This entails two key points. Firstly, we aim to discover a sufficiently effective single-modal data compression algorithm capable of adequately extracting features from both modalities. Secondly, we aim to determine a suitable fusion approach to sufficiently integrate the features from both modalities. Therefore, this chapter will introduce three VAE-based compression models, i.e., [5], [3], and [4]. More specifically, the single-modal architecture is presented in Section 2.4.1. Our approach later in the thesis will then be to extend this single-modality architecture with multimodal properties based on the architectures covered in Sections 2.4.2 and 2.4.3.

2.4.1 LIC-TCM

In this work, the foundational architecture that will be used is called Lossy Image Compression with Transformer-CNN Mixture architectures (LIC-TCM) introduced in [5]. In the paper, the authors propose an efficient parallel Transformer-CNN Mixture block with a controllable complexity to incorporate the local modeling ability of CNN and the non-local modeling ability of transformers to improve the overall performance of image compression models. Furthermore, the authors propose a channel-wise entropy model that incorporates a novel parameter-efficient SWIN transformer-based attention module (see section 2.1.4.2) which the authors have coined as the SWAtten module.

According to the authors, the proposed architecture achieves state-of-the-art results, something that we have verified. A reconstructed sample provided by the authors of LIC-TCM can be seen in Figure 12.

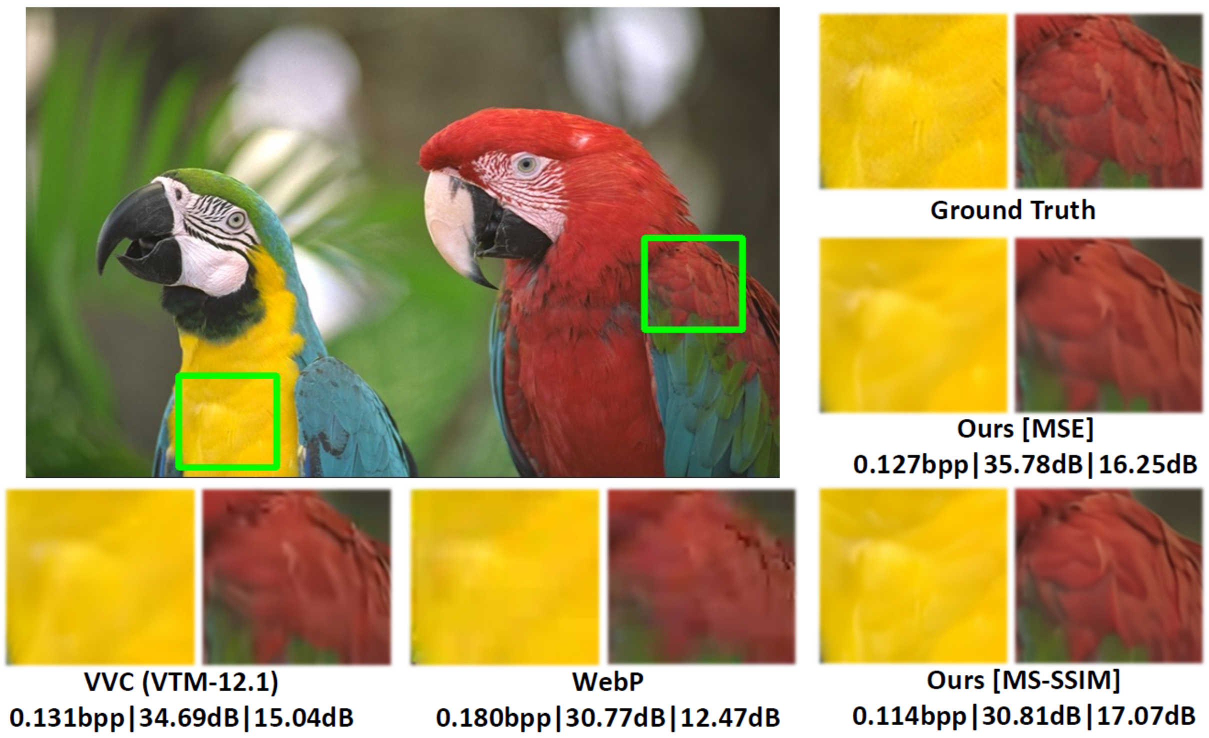


Figure 12: Reconstruction sample of kodim23 of the Kodak dataset [29]. Comparisons are made to illustrate the superior performance of LIC-TCM, where VVC and WebP are two conventional image compression methods. The quantitative results are evaluated by bitrate(bpp) | PSNR(dB) | MS-SSIM(dB) (The reader may refer to section 3.4 for more information on these metrics). Reprinted with permission from authors of [5]. No change has been made to the figure.

Considering their results, the transparent architecture, and open-source code[1], LIC-TCM is the foundational architecture that will be extended to include multimodal properties.

In this section, an overview of the architecture will be presented in the following subsections. It will cover both the image compression architecture and the entropy model architecture. Finally, the loss function of LIC-TCM will be provided in Subsection 2.4.1.4.

The architecture of LIC-TCM is quite complex, encompassing many different machine learning modules and strategies. *Therefore, only some key aspects will be shared and the reader is encouraged to read the paper itself for more details.* Initially, the two major blocks, the image compression architecture and the entropy model with a hyperprior architecture will be covered.

[1]Code can be found here: https://github.com/jmliu206/LIC_TCM

2.4.1.1 Image compression architecture The architecture of the image compression model consists of two blocks, the encoder g_a and the decoder g_s. The encoder consists of three analysis modules, each containing (in order) one residual block with stride (RBS) of two on the first convolution and a Transformer CNN Mixture (TCM) block. An overview of the RBS block can be seen in Figure 13 and an overview of the TCM block can be seen to the right in Figure 14. These analysis modules are sequentially placed and end with one final convolutional layer, i.e., given an image input $\boldsymbol{x}$ to g_a, an encoded output $\boldsymbol{y} = g_a(\boldsymbol{x})$ is produced.

Each TCM block contains two submodules that only differ in one aspect; The first submodule (Stage I) uses a window-based multi-head self-attention (W-MSA) and the second submodule (Stage II) uses a shifted window-based multi-head self-attention (SW-MSA) [16]. Besides this, both contain the same structure and combine ideas such as CNNs and residual networks. It is at the TCM block where most heavy feature extractions are done.

The decoder g_s follows a similar but inverse process of g_a. It also consists of three similar analysis modules, but instead called synthesis modules, and instead of using RBS, it uses a residual block with sub-pixel upsampling with a factor of two on the last convolution (RBU). An overview of the RBU block can be seen in Figure 13.

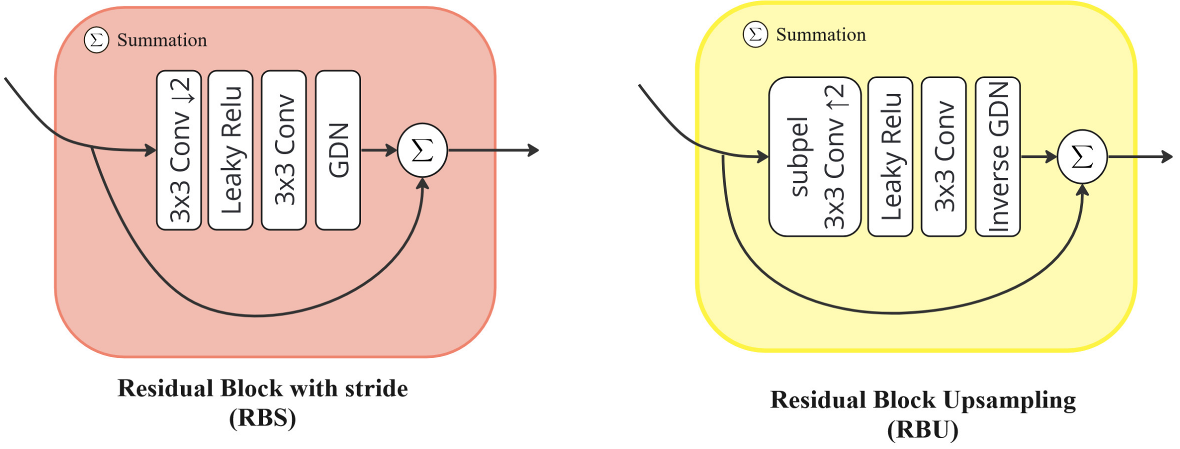


Figure 13: Overview of the RBS- and RBU block.

Again, these analysis modules are sequentially placed and for g_s, it ends with a subpel convolutional layer. To see implementations for subpel 3x3 Conv, RBS and RBU, visit the library by CompressAI in `compressai.layers` [30]. Note that the GDN module stands for generalized divisive normalization which is a parametric nonlinear transformation that is well-suited for image compression [31], [32]. So given a latent representation $\bar{\boldsymbol{y}}$, g_s produces the reconstructed output $\hat{\boldsymbol{x}}$. An overview of the architecture can be seen in Figure 14.

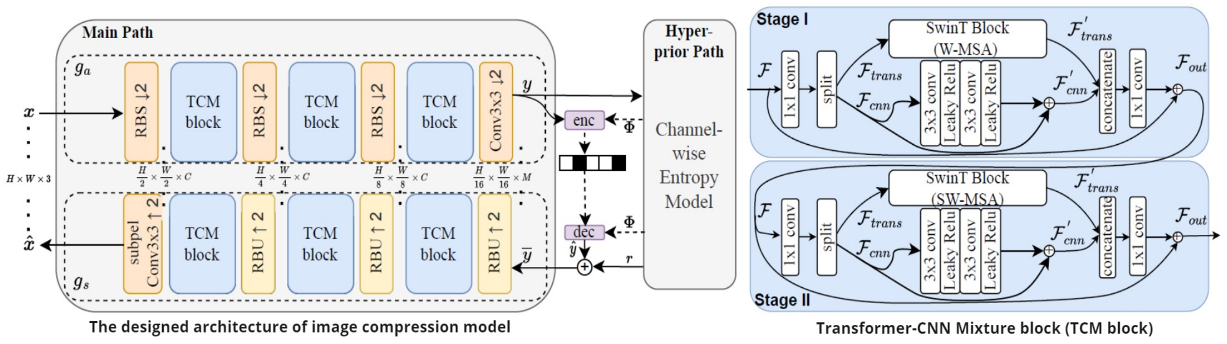


Figure 14: Overall architecture of the LIC-TCM model as provided by [5]. On the left is the image compression model, consisting of the encoder- (g_a) and decoder (g_s) network. Next to the image compression model is the entropy model which improves the entropy of the input tensor y. On the right is an overview of the TCM block. Note how the block is divided in two different stages and only differing in the "SwinT Block". In the first stage, a Window Multi Self-Attention module (W-MSA) is used and in the second stage a Shifted Window Multi Self-Attention module (SW-MSA) is used. Reprinted with permission from authors of [5]. No change has been made to the figure.

2.4.1.2 Entropy model architecture

The entropy model architecture is based on the work of Minnen et al. [33], where they introduced a channel-wise autoregressive entropy model for learned image compression. The idea is to split the latent tensor along the channel dimension into N roughly equal-size slices, and conditions the entropy parameters for each slice on previously decoded slices. This way, the entropy model will achieve superior processing performance. Liu et al. [5] took it a step further and a reduced the amount of slices, from 10 to 5, and introduced the SWAtten module to reduce complexity within the entropy model.

Otherwise, Liu et al. follow a similar structure, where the hyperprior uses a conditional Gaussian model parameterized by both scale and mean and each slice is conditioned on the hyperprior and any prior decoded slices. Furthermore, each slice uses a latent residual prediction (LRP) to reduce quantization error. Since y must be quantized before compression, it inevitably leads to residual error in the latent space $r = y - Q[y]$ that manifests as extra distortion when reconstructed as $\hat{x}$. An LRP is a module with the goal of reducing this residual error. An overview of the entropy model architecture can be seen in Figure 15.

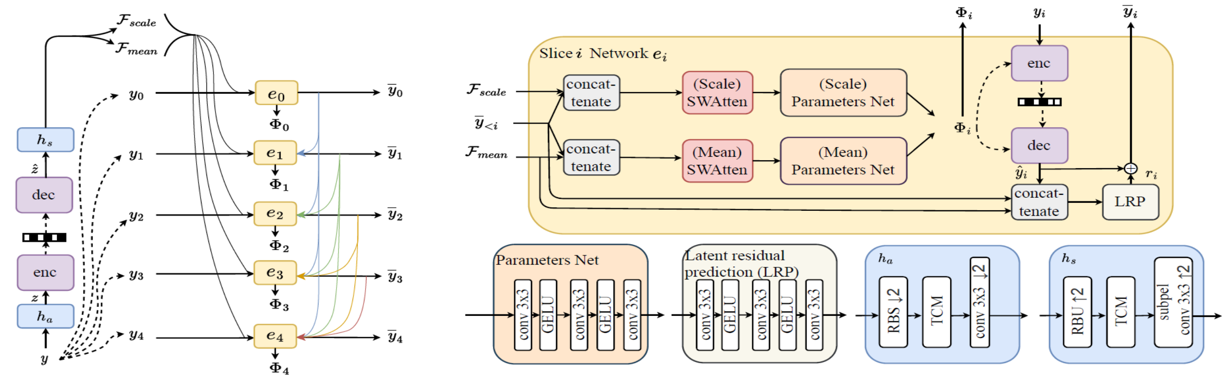


Figure 15: Entropy model architecture used in LIC-TCM as provided by [5]. The important take-aways from this figure are on the left, where the input tensor is split into five slices and y is passed through the entropy model network (h_a and h_s). Further details of this figure is explained in [5]. Reprinted with permission from authors of [5]. No change has been made to the figure.

The output of h_a, $\boldsymbol{z}$, is fed to an entropy bottleneck layer used to model $\boldsymbol{z}$ itself. It is a fully factorized, unconditional entropy model, implemented by CompressAI [30] in `compressai.entropy_models`, more specifically, `compressai.entropy_models.EntropyBottleneck`. On the other hand, in the latent space of $\boldsymbol{y}$, the estimation of entropy is done by a conditionally Gaussian probability density model layer, implemented by CompressAI in `compressai.entropy_models.GaussianConditional`.

2.4.1.3 Encoding and Decoding architecture The encoding and decoding at any compression or decompression step is entropy coded with the ranged variant of asymmetric numeral systems (rANS) [26]. It is implemented by the CompressAI [30] library in `compresssai.ans`, using `compressai.ans.RansEncoder` and `compressai.ans.RansDecoder`.

2.4.1.4 Loss function The authors propose the overall loss function as a Lagrangian multiplier-based rate-distortion optimization. The loss is defined as

$$\begin{aligned} \mathcal{L} &= \mathcal{R}(\hat{\boldsymbol{y}}) + \mathcal{R}(\hat{\boldsymbol{z}}) + \lambda \cdot \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) \\ &= \mathbb{E}[-\log_2(p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}))] + \mathbb{E}[-\log_2(p_{\hat{\boldsymbol{y}}|\boldsymbol{\psi}}(\hat{\boldsymbol{z}}|\boldsymbol{\psi}))] + \lambda \cdot \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}), \end{aligned} \tag{17}$$

where a factorized density model ψ is used to encode quantized $\hat{\boldsymbol{z}}$. Additionally, λ controls the rate-distortion trade-off and $\mathcal{R}(\cdot)$ and $\mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}})$ denotes the bitrate and distortion term, respectively. Pay also attention to its close similarity to (14). The distortion term is usually calculated by Mean Squared Error (MSE) loss or Multi-Scale Structural Similarity Index Measure (MS-SSIM) loss. Depending on the choice of MSE or MS-SSIM, the lambda values vary. The authors propose for their experiments to use

$\{0.0025, 0.0035, 0.0067, 0.0130, 0.0250, 0.0500\}$ and $\{3, 5, 8, 16, 36, 64\}$ for MSE and MS-SSIM respectively (larger value meaning higher quality and less compression). However, the reader will see that the loss function is subject to change to account for multimodal properties (see (24)).

2.4.2 Convolution-based Multimodal Fusion

The architecture proposed in the paper [3] provides a reasonable strategy to fuse two modalities. In this paper, the authors argue that there exists some shared information between RGB images and depth images. Although the two modalities express the same scene in different ways, they can contain overlapping, complementary, or other forms of relevance in semantic information. Therefore, it is possible to extract structure priors from corresponding RGB images to aid the compression of depth images. In [3], a novel architecture is proposed, consisting of two analysis pipelines for RGB images and depth images, respectively as well as various fusion modules. One can view it as a duplication of a single-modal algorithm, but with added fusion between the modalities. The overall framework can be seen in Figure 16. The analysis block can be any codec that captures features from the input image.

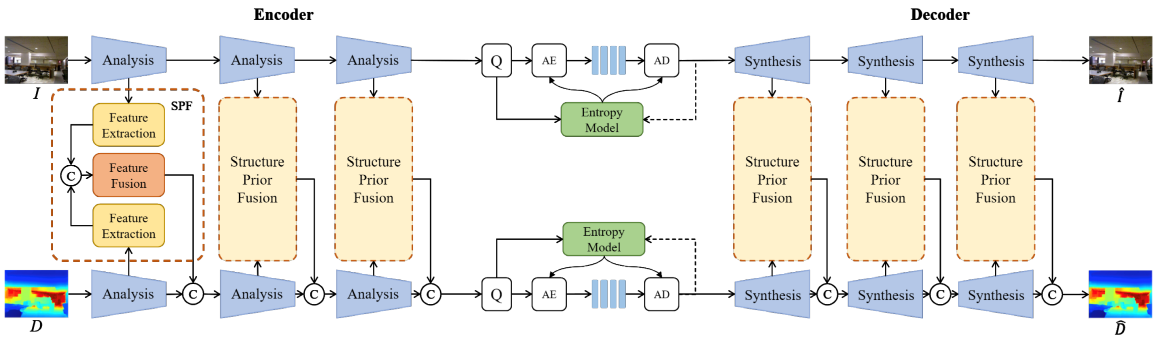


Figure 16: Overall architecture as provided by [3].

To achieve information exchange, the Structure Prior Fusion (SPF) module is employed to extract features from both RGB and depth images and subsequently fuse them. Specifically, this module is positioned after each analysis block or each synthesis block, initiated by a convolutional layer for feature extraction. Following this, the two tensors, housing RGB and depth information, respectively, are concatenated depth-wise. The Enhanced Spatial Attention Block (ESA) is implemented on the concatenated tensor to emphasize the region of interest. Firstly, a 1×1 convolutional layer is employed to reduce the dimensionality of the tensor, rendering the block lightweight. Subsequently, a convolutional layer with stride is utilized to expand the receptive field, enhancing the block's capability. The third component of the ESA block comprises a max-pooling layer and a series of convolutional layers aimed at feature extraction and dimension reduction. Finally, the combination of an up-sampling layer and a 1×1 convolutional layer is used to restore the shape of the input tensor to its original form within this block. Figure 17 illustrates the detailed structure of the SPF module and the ESA block. Note that the 3×3 convolutional layer after the summation is not part of the ESA block.

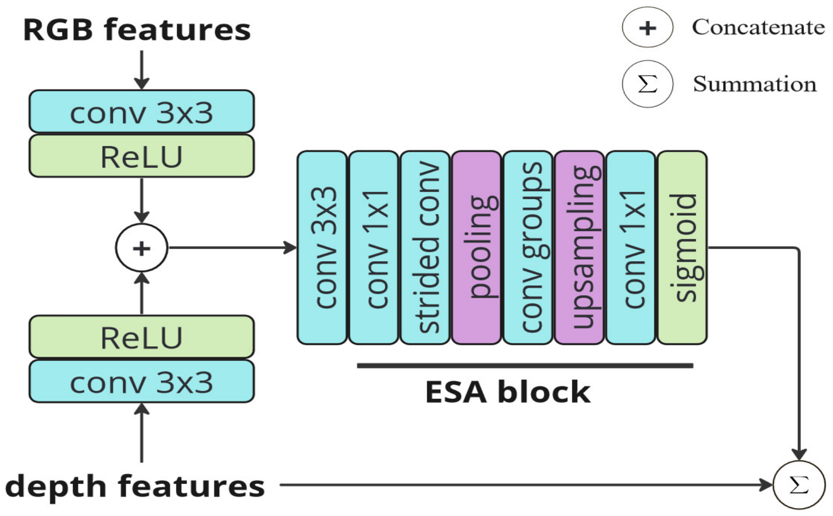


Figure 17: Detailed structure of the SPF Module and the ESA Block.

2.4.3 Attention-based Multimodal Fusion

In [4], the authors Gnutti et al. propose a novel architecture that utilizes the LiDAR depth map as supplementary information to achieve superior results. The authors claim that their results yield an average PSNR gain of 0.83 dB and an average bitrate reduction of 16% as compared to the baseline (the concepts of PSNR and bitrate will be introduced in detail in section 3.4). The work is an extension of the paper [34], but omits the Region-Of-Interest (ROI) mask and uses the LiDAR depth map instead. Finally, the work in [34] with prompts is inspired by a more well-noticed paper called "*Visual Prompt Tuning*" [35].

In [34], the authors attach a prompt generative network to the encoder and decoder, p_a and p_s, respectively. The input of p_s is a concatenated tensor of three components: ROI mask $M_R \in \mathbb{R}^1$, lambda map ($M_\lambda \in \mathbb{R}^1$) and the input image ($x \in \mathbb{R}^3$). The ROI mask is populated with values in $[0, 1]$, which functions as weights to the pixels. The lambda map is populated with the distortion trade-off λ (see Section 2.4.1.4) but should be in $[0, 1]$. In a similar manner, p_s accepts the image latent vector $\hat{y}$ and a downscaled lambda map $\hat{M}_\lambda \in \mathbb{R}^{1 \times \frac{H}{16} \times \frac{W}{16}}$ such that it matches the spatial resolution of the latent vector $\hat{y}$.

The prompts generated by these networks are fed to, what the authors coined as, prompted SWIN transformer block (P-STB). These P-STB modules accept both prompt and image as input and are tokenized for the SWIN transformers. The prompt window partition generated by the p_a network is lower than the image partitions to reduce complexity. To get an overview of the generated tokens and how the image tokens and prompt tokens interact cooperatively, see Figure 18.

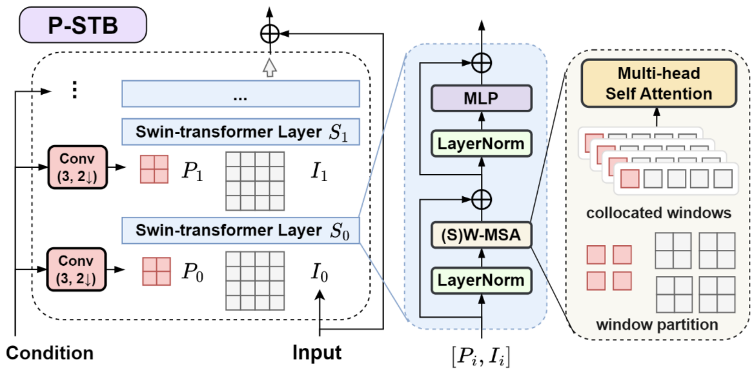


Figure 18: Overview of a P-STB module as provided by [34]. Notice how a prompt token (P_i) has reduced amount of tokens compared to the image tokens (I_i). No alteration has been made to this figure. Licensed by CC BY-NC-ND 4.0.[2]

As previously stated, [4] is an extension of [34], but with two significant distinctions. Firstly, the ROI mask is replaced with a depth map $M_D \in \mathbb{R}^1$ and secondly, two new prompt generative networks are attached to the encoder l_a and decoder l_s. According to the authors, these prompts allow the model to leverage the LiDAR information effectively for both the encoding and decoding processes. Besides these two new additions M_D and l_a and l_s, the architecture remains the same. See the updated architecture in Figure 19.

[2]https://creativecommons.org/licenses/by-nc-nd/4.0/, through https://arxiv.org/abs/2401.06517

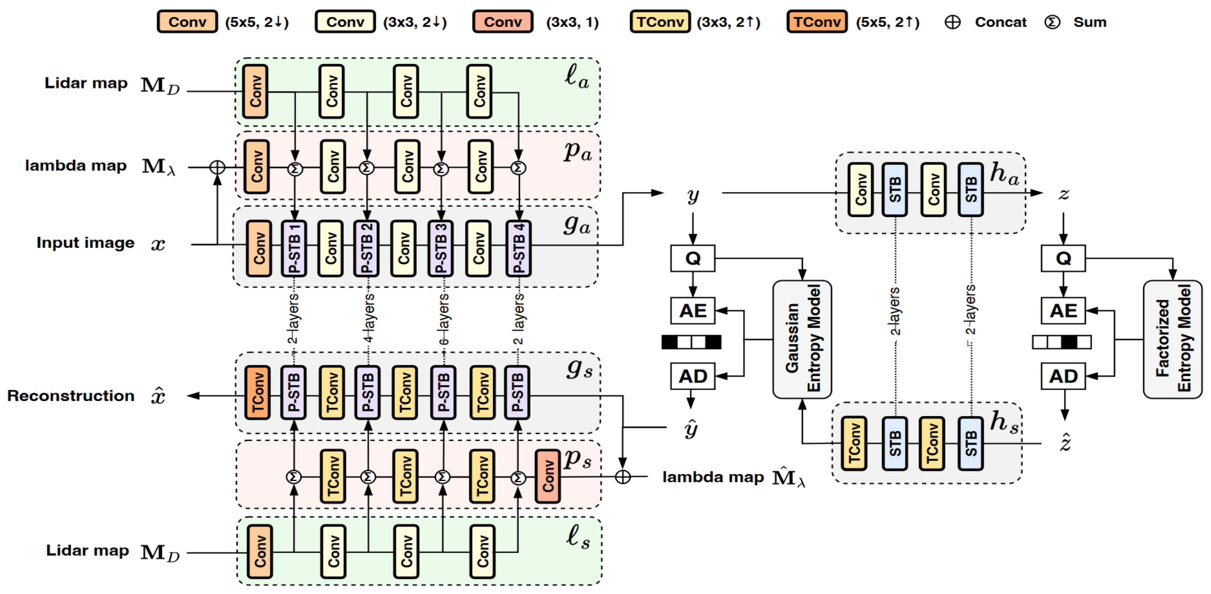


Figure 19: Overall architecture as provided by [4]. No alteration has been made to this figure. Licensed by CC BY 4.0.[3]

[3]See footnote 2.

3 Methodology and Implementation

As previously stated, LIC-TCM (see Section 2.4.1) will be the foundational architecture that will be extended to support multimodal properties. In this work, three different variants of LIC-TCM extensions will be implemented, trained, and evaluated. The implementations are based on the architectures provided in Section 2.3 and the details of these implementations can be found in Section 3.1. The chosen datasets along with details and reasoning of why it was chosen can be found in Section 3.2. The evaluation of the trained models will be evaluated according to the metrics provided in Section 3.4. Finally, in Section 3.5 the configurations surrounding the experiments will be provided, ranging from specific hyperparameters to hardware used.

3.1 Architectures

In this section, we propose three novel architectures. In the first variant, a straightforward architecture is implemented where the two input modalities are simply concatenated before entering the network, achieving early fusion. In the two remaining architectures, we employ two different fusion approaches as proposed in [3] and [4], respectively. However, the analysis blocks are selected from LIC-TCM [5] to perform feature extraction, given its state-of-the-art performance. All the above-mentioned models are introduced in Section 2.4.

To clarify, the first architecture (3.1.1) accepts both RGB and depth as joint input modalities and can output either RGB or depth for reconstruction, depending on the configuration. The other two networks also accept RGB and depth as joint input modalities, but differ in their outputs: one architecture (3.1.2) targets only depth for reconstruction, while the other (3.1.3) targets only RGB. The rationale behind these design choices is discussed further in 5.2.

3.1.1 4-channel LIC-TCM

At a very simple level, an investigation is done to see how the original LIC-TCM performs if the input channel is extended from 3 (RGB) to 4 (RGB plus depth). The original architecture will otherwise remain the same, see Figure 20. The output of the architecture will be three-channel RGB or single-channel depth, and the model named, 4 Channel input LIC-TCM, will be trained to improve this output. Based on different targets (i.e., either RGB or Depth), we call these slightly different architectures 4-channel LIC-TCM (RGB target) and 4-channel LIC-TCM (Depth target), respectively. Note that in the following text, we may refer to “4-channel” as “4C” for short. While a very simple architecture, this architecture is investigated because of its effortless implementation cost.

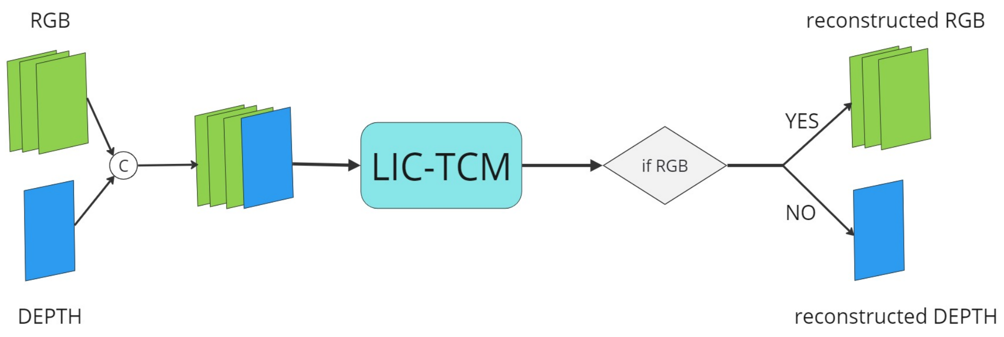


Figure 20: Architecture of the 4C LIC-TCM model.

3.1.2 Convolution-based Multimodal-LIC-TCM

In this architecture, we use the combination of Transform-CNN Mixture (TCM) and residual block with stride (RBS) blocks from LIC-TCM model as analysis block considering its ability to apprehend both local patterns and non-local information [5]. The details of the TCM block can be found in Figure 14. Hence, the basic architecture of LIC-TCM is kept. On top of that, a separate pipeline is added to the *Main Path* in the original LIC-TCM, which is the RGB compression pipeline. Similarly, it has the same analysis block and is used to learn color information from the RGB modality. In other words, there are two g_a in the encoder, namely $g_{a\text{RGB}}$ and $g_{a\text{depth}}$, which mean analysis blocks for the RGB inputs and the depth inputs respectively. Moreover, SPF modules (see Section 2.4.2 and Figure 17) are applied to extract and fuse the features from the two modalities. Different from the codec used in [3], we only implemented two phases of fusion rather than incorporating six SPF modules, and both of them are placed before the encoder. The reason is that the SPF block is relatively resource-intensive, it is advisable to restrict its quantity to prevent the model from becoming excessively large. Note that in our work, we use this fusion strategy for depth compression, while [3] leverages the depth map to assist RGB compression, which is the other way around.

To conclude, this architecture targets multimodal data compression by using LIC-TCM as its backbone and incorporating a convolution-based fusion strategy. Therefore, we name this new architecture Convolution-based Multimodal-LIC-TCM, abbreviated as Convolutional MM-LIC-TCM. The overall architecture of the Convolutional MM-LIC-TCM model is illustrated in Figure 21

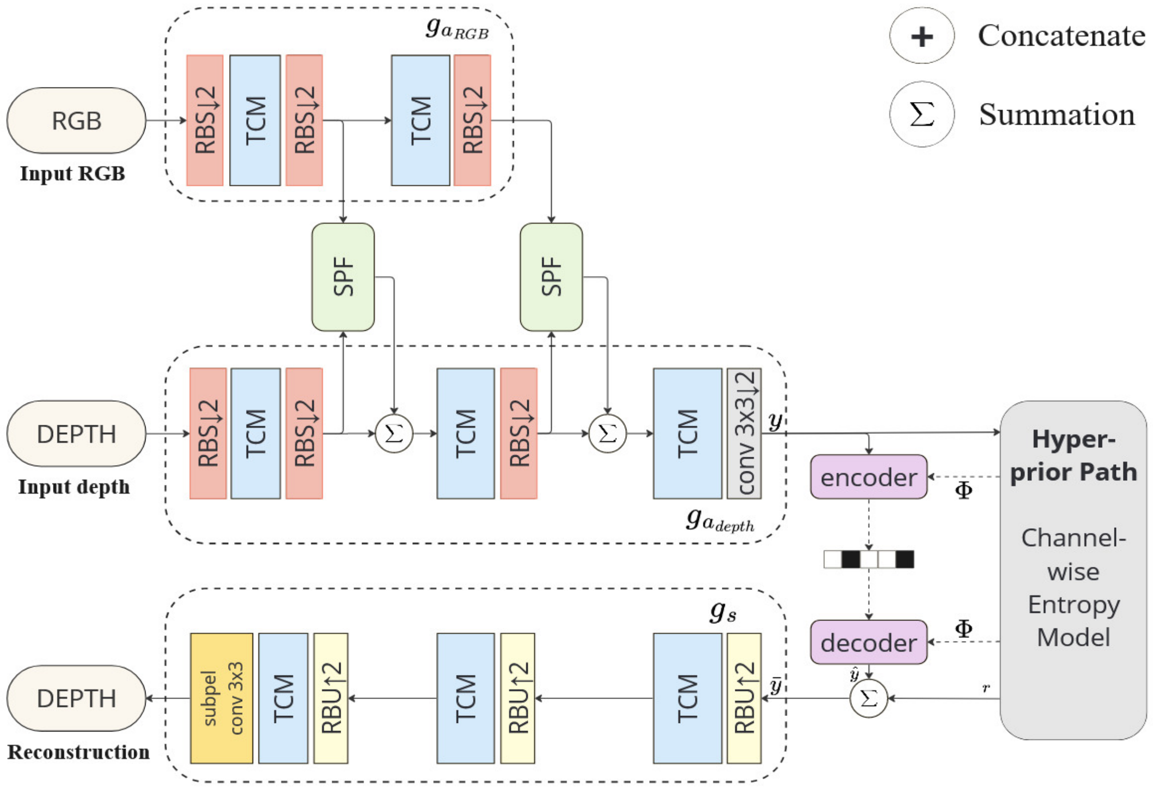


Figure 21: Architecture of the Convolutional MM-LIC-TCM model.

To make it clear, Table 1 summarizes the full names and the main functions of the different blocks in Figure 21. The reader can find a more detailed description in Section 2.4 and in the original paper of LIC-TCM [5].

Table 1: A short description of the blocks in Figure 21.

Block	Full Name	Description
RBS	Residual Block with Stride	Feature Extraction & Downsampling
TCM	Transformer-CNN Mixture	Feature Extraction
SPF	Structure Prior Fusion	Fusion
RBU	Residual Block Upsampling	Upsampling

3.1.3 Attention-based Multimodal-LIC-TCM

In this architecture, we propose an updated LIC-TCM architecture that implements the two prompt generative networks, p_a and l_a, and attaches it to the encoder, g_a. Additionally, we implement the depth map and lambda map, M_D and M_λ, respectively, and connect these to the prompt generative networks, see Figure 23. Instead of using P-STBs as analysis modules, we use LIC-TCM's TCM modules instead which are augmented with the capability of accepting both image tokens and prompt tokens. The prompt token and the image tokens are separately fed to their own layernorm, thereafter the prompt token is prepended to the image tokens and passed to the attention module. This augmented TCM module is referred to as Prompted TCM and an overview of our design is seen in Figure 22. Note that for TCM Stage I, W-MSA is used and for TCM Stage II, SW-MSA is used. Considering the way it fuses the two modalities before being used in the attention module, we call this architecture Attention-based Multimodal-LIC-TCM, abbreviated as Attention MM-LIC-TCM.

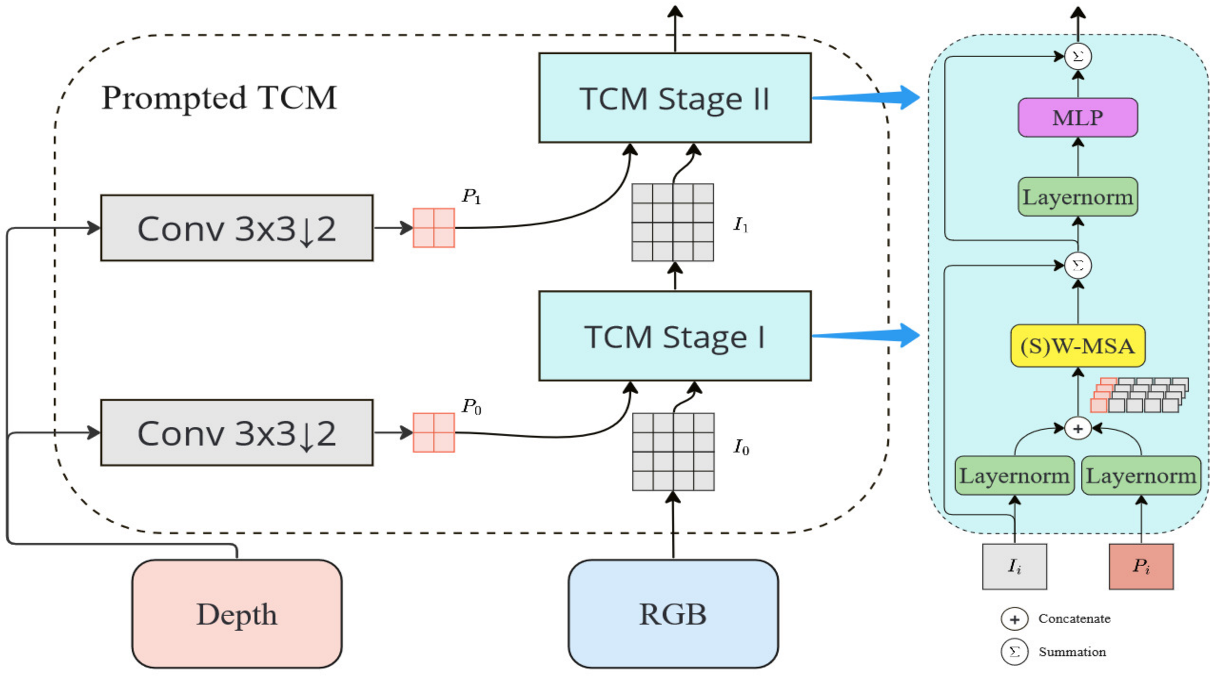


Figure 22: Overview of the Prompted TCM module.

The careful reader might notice that p_s and l_s (the lambda map and lidar map networks on the decoder side of the image compression model) are omitted from the architecture. This was a choice by design, supported by three aspects. Firstly, the inclusion of p_s and l_s adds time complexity to the network—an already quite complex architecture. This would affect performance in terms of encoding and decoding speed and model size. Secondly, this thesis work is time-constrained, and integrating prompt tokens to the SWIN transformers is a time-consuming process to implement and integrate; hence, we chose to omit it. Thirdly, the authors in [34] made an experiment of excluding LiDAR information at the decoder and saw comparable results to when the entire architecture as in Figure 19 was used. However, we will discuss the possible advantages of such architectural decisions in Section 5.

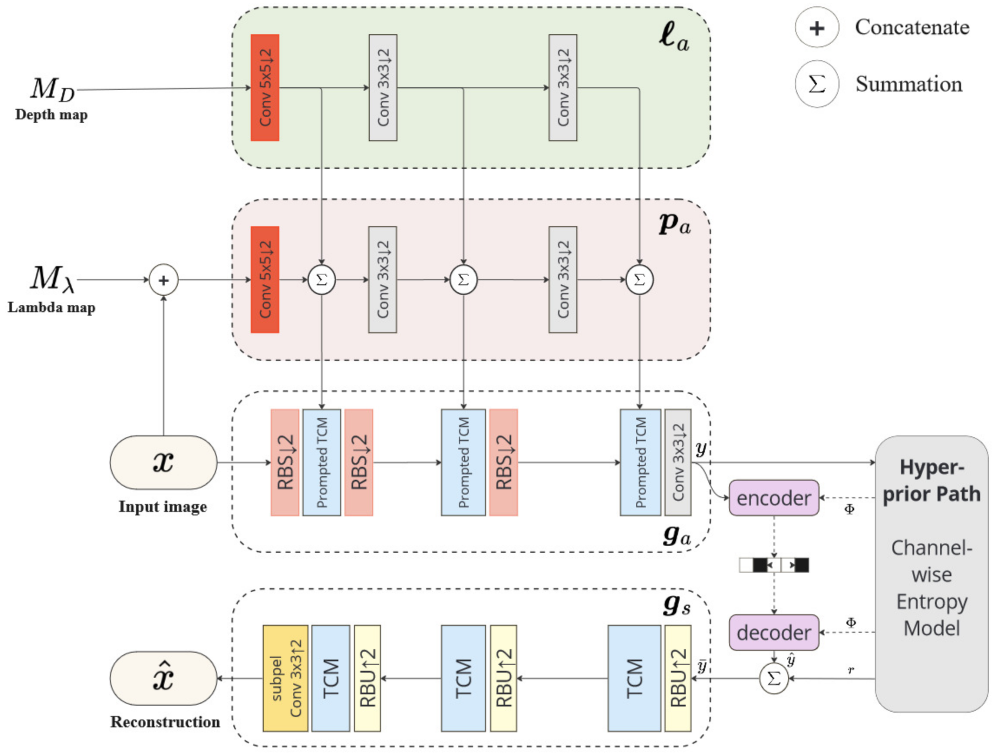


Figure 23: Architecture of the Attention MM-LIC-TCM model.

3.2 Training Dataset

The dataset used is an indoor LiDAR-RGBD scan dataset from the Redwood-2017 dataset [36]. It contains over 120 000 RGB-D pairs reaching 285 GB uncompressed. The Redwood-2017 dataset consists of five different subsets from different RGB-D sequences taken in different environments. These five environments are named *Apartment*, *Bedroom*, *Boardroom*, *Lobby* and *Loft*. For this thesis work, *Apartment*, *Bedroom*, *Boardroom* and *Lobby* is used for training (∼100k RGB-D pairs) and *Loft* is used for validation (∼20k RGB-D pairs). The LiDAR scan data was collected using a FARO Focus 3D X330 HDR scanner with an operating range of 0.6m to 330m. At a distance of 10 meters, its ranging accuracy is 0.1 millimeters. For each RGB-D sequence a color image is aligned with the corresponding depth image. Each color image is stored as 8-bit JPG and each depth image is stored as 16-bit PNG, where the latter's pixel values represent depth in millimeters. For each frame, the resolution is 640×480. Finally, the focal length is 525 for both axes and the principal point is (319.5, 239.5).

Furthermore, documentation exists to reconstruct and create 3-dimensional renderings of the Redwood RGB-D pairs with the aid of Open3D [37]. This is a nice feature in order to provide qualitative results for the reader, especially in the case of compressing depth information. More information on the dataset can be found on the official website. [4]

3.3 Test dataset

The testset is based on two sceneries from the Augmented ICL-NUIM Dataset, living room 1 and office 2 [38] [39]. This testset will be used for evaluating the performances of our implementations. This dataset belongs to a different version of the Redwood family. The original ICL-NUIM aims at benchmarking RGB-D and has noiseless depth images and therefore is a good match as a test dataset. A total of 40 RGB-D pairs (each frame with resolution 640x480) were manually extracted and chosen as a test set. An overview of the test data set can be seen in Appendix A.

3.4 Evaluation

To evaluate the performance of a lossy learned image compression model, multiple metrics have to be taken into consideration. Ranging from the encoding and decoding time to more advanced metrics that measure how good the image compression is in terms of quality and size reduction. In this section, the different metrics that are used to evaluate the implementations will be covered.

3.4.1 PSNR

In image compression, peak signal-to-noise ratio (PSNR) is a ratio between the maximum possible value in bit depth and the pixel value of distorting noise versus its original pixel value representation. Overall, PSNR is a common metric in signal processing and image/video quality assessment, and in the latter case is typically used to measure the quality of reconstruction of lossy compression codecs. The reason is that lossy compression usually introduces errors such as visual artifacts.

The PSNR expressed in decibels (dB) with MSE can be defined, with the aid of simple logarithmic rules, as

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{P_{\text{MAX}}^2}{MSE}\right) \\ &= 20 \cdot \log_{10}(P_{\text{MAX}}) - 10 \cdot \log_{10}(MSE), \end{aligned} \tag{18}$$

where P_{MAX} is the dynamic range (i.e., the difference between the maximum and minimum allowed values) and MSE is the mean square error.

In the context of lossy image compression additional constraints can be given to (18). For example, for some given 8-bit RGB image, meaning 24-bit depth (8 bits per color channel),

[4]http://redwood-data.org/indoor_lidar_rgbd/index.html

then P_{MAX} will be set to 255 ($2^8 - 1 = 255$). Furthermore, when applying MSE, it will be the sum of all squared differences for each channel (meaning R-channel, G-channel and B-channel) divided by image size ($H \times W$) and by three. Additionally, if for example PyTorch's `ToTensor()` is used to convert the input image to a tensor then the tensor will be in the range `[0.0, 1.0]`, meaning that P_{MAX} equals one, reducing (18) to

$$PSNR = -10 \cdot \log_{10}(MSE).$$

As a final comment, PSNR should be used cautiously in image/video quality assessment. The PSNR metric is *not* an objective metric and does not reflect the human's subjective perceived quality of an image. However, it can be a good quantitive metric to indicate that the reconstruction *might* be good. Even with this in consideration, PSNR is still often widely used in terms of image quality assessment.

3.4.2 Bits Per Pixel

Bits per pixel (bpp) is a metric used to quantify the amount of information stored in a pixel. In image compression, it serves as a measure of how much an image is compressed. For a standard RGB image, each pixel is typically represented by three color channels: red, green, and blue. In many scenarios, the color depth of each channel is 8-bit. Hence, the bpp of such an image would be 24, reflecting the combined bit depth of all three channels.

In LIC-TCM, bpp is calculated by the sum of the compressed bitstreams in the latent space of the image compression architecture and in the latent space of the entropy model architecture, and divided by the total number of pixels in the image. In practice, these bitstreams are represented in a *y_string* and *z_string* which can be decoded by the decoder of the image compression decoder and the entropy model decoder, respectively.

It is important to note that all the depth images utilized in our work are formatted as 16-bit one-channel grayscale images. As a result, the compressed depth image should inherently have a relatively lower bpp value compared to RGB images, given the lower bpp of the original images.

3.4.3 MS-SSIM

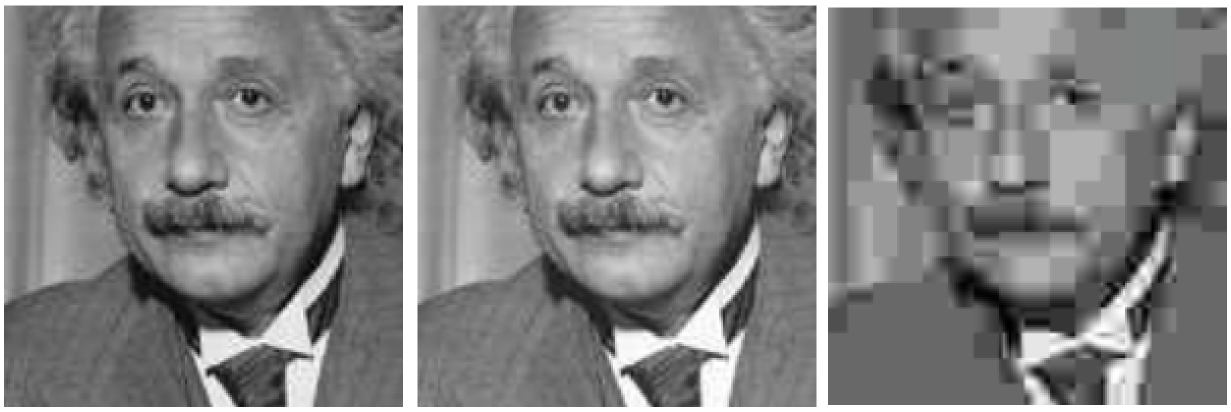
Multi-Scale Structural Similarity Index Measure (MS-SSIM) is another common metric for image/video quality assessment [40]. Whereas PSNR focuses more on the absolute error between the pixels, MS-SSIM measures the difference between three properties of the pixels and is an extension of the previous work of SSIM [41] (Structural Similarity Index Measure).

Given two image patches extracted from the same spatial location, say $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ consisting of N pixels each, SSIM measures the luminance $l(\cdot,\cdot)$, contrast $c(\cdot,\cdot)$ and structure $s(\cdot,\cdot)$, and the product of these factors gives SSIM. In other words, the general form of SSIM between a signal $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ is defined as

$$\mathrm{SSIM}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = [l(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\alpha} \cdot [c(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\beta} \cdot [s(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\gamma}, \tag{19}$$

where α, β and γ are parameters to define relative importance. The reader can refer to [41] to read about the details of the functions within (19).

In many scenarios, SSIM is regarded as a more appropriate metric for quantifying the disparity between a reconstructed image and the original image. The paper in [41] contends that, in comparison to Mean Square Error (MSE), SSIM excels at capturing the structural information of the image. This assertion is elaborated in [41]. This choice of metric is rooted in the desire to align with the mechanisms of the human visual system, reflecting the aim of mimicking human behavior in assessment criteria. An example will be given below to illustrate the effectiveness of SSIM, see Figure 24. Between the two distorted images, people tend to think the image in the middle is closer to the original image, while the MSEs are the same. A high SSIM value indicates that the Figure 24b possesses a similar structure to the original one.



(a) MSE=0, SSIM=1. **(b)** MSE=309, SSIM=0.987. **(c)** MSE=309, SSIM=0.580.

Figure 24: The images of Einstein with different kinds of distortions. (a) Original image. (b) Luminance shift. (c) JPEG compression.

MS-SSIM extends (19) by iteratively applying a low-pass filter and downsampling the filtered image by a factor of 2. Given a maximum scale of M, the updated definition in [40] is given as

$$\text{MS-SSIM}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = [l_M(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\beta_j} \cdot [s_j(\boldsymbol{x}, \hat{\boldsymbol{x}})]^{\gamma_j}, \tag{20}$$

where again similarly, α_M, β_j and γ_j are parameters to define relative importance. For a clear system diagram see Figure 1 in 20.

3.4.4 Comprehensive Similarity

In [42], the author introduces a novel loss function where pixel-wise difference, gradient difference and structure similarity term are combined together to measure the distortion

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \gamma L_{depth}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + L_{grad}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + L_{ssim}(\boldsymbol{y}, \hat{\boldsymbol{y}}). \tag{21}$$

The first term L_{depth} measures the pixel-wise L_1 loss while the second term calculates the gradient difference between the two images. In the following equations, g_x and g_y represent the gradient differences in x and y directions respectively, between the two depth images y and $\hat{y}$

$$L_{depth}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{p}^{n} \mid \boldsymbol{y_p} - \hat{\boldsymbol{y}}_{\boldsymbol{p}} \mid \tag{22}$$

$$L_{grad}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{p}^{n}(\mid g_x(\boldsymbol{y}_p, \hat{\boldsymbol{y}}_p) \mid + \mid g_y(\boldsymbol{y}_p, \hat{\boldsymbol{y}}_p) \mid). \tag{23}$$

Moreover, the third term simply estimates the structure similarity of the images (see 19). The γ parameter attached to the L_1 loss is set to adjust the weight. The effectiveness of this comprehensive metric is verified by [42] and [22]. Given this comprehensive similarity metric, it was used as a distortion term to better train and validate the performance of the architectures with depth as the target output. However, for our implementations, we updated the final term to use MS-SSIM instead of SSIM. The final equation is given as

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \gamma L_{depth}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + L_{grad}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + L_{ms-ssim}(\boldsymbol{y}, \hat{\boldsymbol{y}}). \tag{24}$$

3.5 Experimental Setup

In this paper, six different architectures (see Section 3.1) are implemented with three different configurations each. For each architecture, the most significant configuration parameter is the lambda hyperparameter (see Section 2.2.2) because of its direct impact on the bitrate - quality trade-off. Secondly, the distortion term varies between the RGB and depth architectures so it is important to clarify these distinctions. Again, we use the single-modal image compression codec LIC-TCM as the baseline for both RGB and depth compression. To get an overview of the different configurations per architecture, see Tables 2 and 3 for RGB and Depth baselines, respectively. The configurations per fusion architecture can be found in Tables 4, 5, 6, and 7, for Attention MM-LIC-TCM, Convolutional MM-LIC-TCM, 4C LIC-TCM (RGB target), and 4C LIC-TCM (Depth target) architectures, respectively. Notably, the depth codec in Convolutional MM-LIC-TCM is initialized with the pre-trained weights from the depth baseline and these weights are frozen to prevent further training. Attention MM-LIC-TCM was not given the same treatment, because the Prompted TCMs conflict with the baseline weights. Thus, Attention MM-LIC-TCM was trained as a whole.

Table 2: Baseline - RGB

Configurations	**Values**
Lambda	0.0025 ; 0.0130 ; 0.0500
Distortion term	MSE

Table 3: Baseline - Depth

Configurations	**Values**
Lambda	0.0001 ; 0.001 ; 0.005
Distortion term	Comprehensive Similarity, see (24) (γ=0.1)

Table 4: Fusion - Attention MM-LIC-TCM

Configurations	**Values**
Lambda	0.0025 ; 0.0130 ; 0.0500
Distortion term	MSE

Table 5: Fusion - Convolutional MM-LIC-TCM

Configurations	**Values**
Lambda	0.0001 ; 0.001 ; 0.005
Distortion term	Comprehensive Similarity, see (24) (γ=0.1)

Table 6: Fusion - 4C LIC-TCM (RGB target)

Configurations	**Values**
Lambda	0.0025 ; 0.0130 ; 0.0500
Distortion term	MSE

Table 7: Fusion - 4C LIC-TCM (Depth target)

Configurations	**Values**
Lambda	0.0001 ; 0.001 ; 0.005
Distortion term	Comprehensive Similarity, see (24) (γ=0.1)

Besides architectural differences and lambda values, all of the configurations share the same experimental setup. The experimental setup is highly similar to LIC-TCM's setup, but for clarity the setup is shared: For training, images from our dataset (see Section 3.2) are randomly chosen and cropped with the size of 256×256 during training. Adam [43] is used as our choice of iterative optimization algorithm for the loss function, with a batch size set to 8, both for validation and training. An initial learning rate is set to $1 \cdot 10^{-4}$. The learning rate is then scheduled by Pytorch's `MultiStepLR()` which will be decreased by a factor of ten at epochs 45 and 48.

The SWIN transformer blocks have window sizes set to 8 for the encoder and decoder, g_a and g_s, respectively, and window sizes are set to 4 in the hyperprior path (h_a and h_s). The channel number C is set to 64 (smallest setting) for all of the architectures to reduce complexity. Finally, the latent channel number of $\boldsymbol{y}$, M, is set to 320 and correspondingly $\boldsymbol{z}$ is set to 192.

The sessions were trained on RTX 3090 GPU and RTX 4090 GPU provided by Ericsson. Additional training sessions were also enabled by resources provided by LUNARC (COSMOS), The Centre for Scientific and Technical Computing at Lund University, using Apptainer/Singularity on A40 and 100 GPU nodes.

4 Results

In this section, a distinction between quantitative results and qualitative results is made. In the quantitative results, only evaluation data is shared in terms of scalar values which are also displayed in line charts. In the qualitative results, some hand-picked reconstructions & renderings of the models are displayed and compared. The results for this section are based on the configurations and experimental setup in Section 3.5 and evaluated on the test set in Section 3.3.

4.1 Quantitative

The quantitative results were produced by loading the trained weights for the given architecture and forwarding the different concatenated RGB-D pairs from the testset. The quantiative results are displayed in terms of PSNR versus bpp and Comprehensive Similarity versus bpp. In general, we want to maximize PSNR and Comprehensive Similarity and minimize bpp. For clearer comparison, the MS-SSIM is converted to $-10\log_{10}(1 - \text{MS-SSIM})$. All forward latency, parameter, and GFLOP data were extracted by using `get_model_profile()` from `deepspeed.profiling.flops_profiler` in the Python-based DeepSpeed library using a RTX 4090 GPU connected to CUDA.

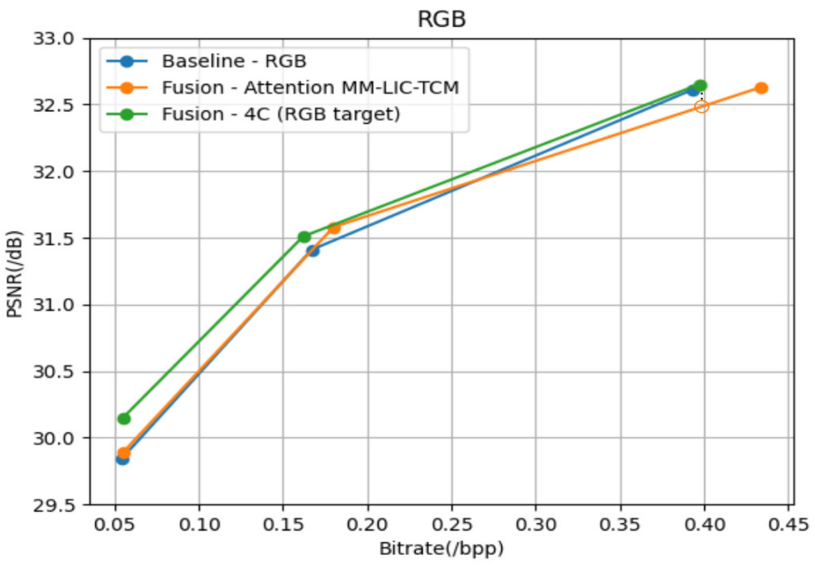


Figure 25: Performance evaluation on the set given in average PSNR and bpp, with lambda configurations (from left to right in the figure): {0.0025, 0.0130, 0.0500}. The non-filled circle is visualized for illustrative purposes.

In Tables 8, 9, and 10 the results for the different lambda configurations per architecture on the evaluation set are given. These tables represent the baseline RGB, 4C LIC-TCM (RGB target), and Attention MM-LIC-TCM architectures, respectively. The PSNR- and bpp values for the tables are visualized in Figure 25. Overall, the fusion architectures outperform the RGB baseline. However, there is an outlier for the Attention architecture for lambda set to 0.0500. Although the other points have a similar bpp, it is difficult to determine how the architectures will perform before-hand, since the lambda hyperparameter uniquely affects the different architectures. By using a slightly lower lambda value for the Attention architecture (0.0400 instead of 0.0500 etc), it hopefully will be closer to the other points and make for a better comparison. Additionally, pay attention to the relative strong performance of the 4C architecture, especially considering its simple low complexity design. Finally, although MS-SSIM values are given in previously mentioned tables, it is not visualized.

Table 8: Baseline - RGB

λ-config	**Average PSNR**	**Average MS-SSIM**	**Average Bit-rate**
0.0025	29.84 dB	12.97	0.054 bpp
0.0130	31.41 dB	15.55	0.167 bpp
0.0500	32.61 dB	17.39	0.393 bpp

Table 9: Fusion - 4C LIC-TCM (RGB target)

λ-config	**Average PSNR**	**Average MS-SSIM**	**Average Bit-rate**
0.0025	30.15 dB	13.21	0.055 bpp
0.0130	31.51 dB	15.36	0.162 bpp
0.0500	32.65 dB	17.32	0.397 bpp

Table 10: Fusion - Attention MM-LIC-TCM

λ-config	**Average PSNR**	**Average MS-SSIM**	**Average Bit-rate**
0.0025	29.89 dB	13.00	0.055 bpp
0.0130	31.58 dB	15.57	0.180 bpp
0.0500	32.63 dB	17.41	0.434 bpp

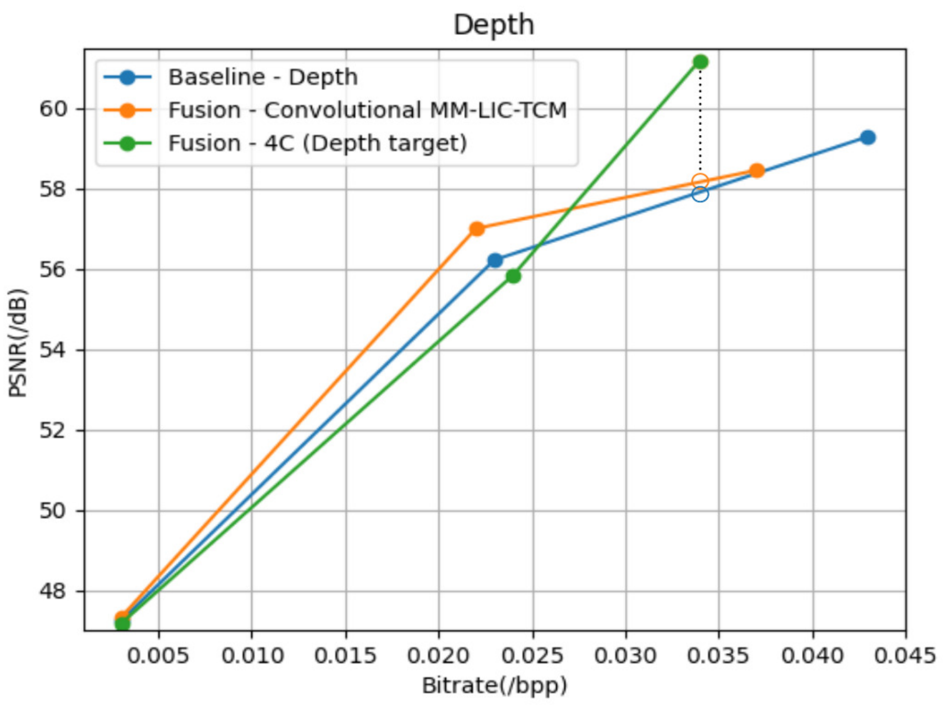


Figure 26: Performance evaluation on the test set given in average PSNR and bpp, with lambda configurations (from left to right in the figure): {0.0001, 0.001, 0.005}. The non-filled circles is visualized for illustrative purposes.

In Tables 11, 12, and 13 the results for the different lambda configurations per depth architecture on the evaluation set are given. These tables represent the baseline depth, 4C LIC-TCM (Depth target), and Convolutional MM-LIC-TCM architectures, respectively. Overall, for lambda equals 0.0001, very similar results were generated. However, discrepancies are unraveled for larger lambdas, especially in the case of the 4C architecture. The evaluation results for these values are directly represented in Figure 26 and Figure 27 for PSNR and comprehensive similarity, respectively.

It can be seen from the figure that our Convolutional MM-LIC-TCM model outperforms the baseline model at all bitrates, while the performance of 4C LIC-TCM is not stable. Overall, the convolutional MM-LIC-TCM demonstrated the best consistent performance.

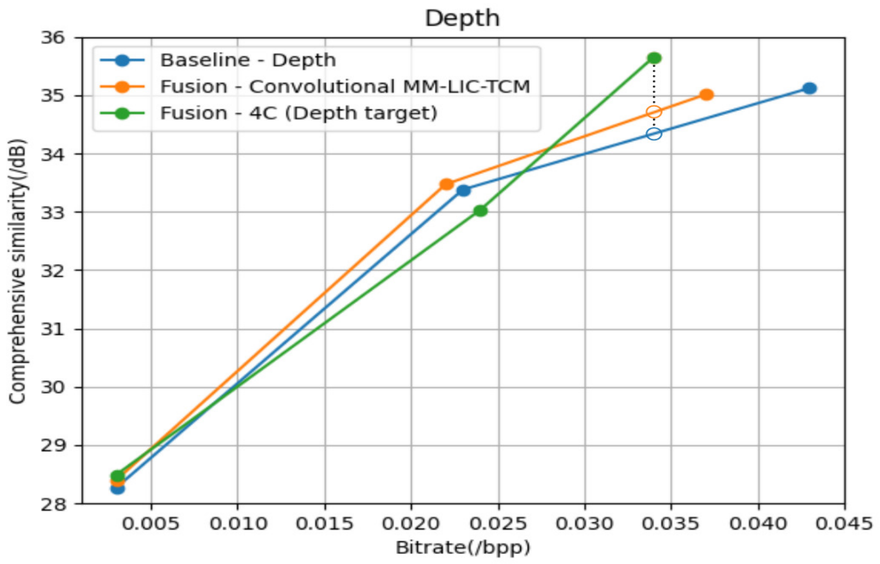


Figure 27: Performance evaluation on the test set given in average comprehension similarity and bpp, with lambda configurations (from left to right in the figure): {0.0001, 0.001, 0.005}. The non-filled circles is visualized for illustrative purposes.

Table 11: Baseline - Depth

λ-config	**Average PSNR**	**Average Comprehensive Similarity**	**Average Bit-rate**
0.0001	47.24 dB	28.27 dB	0.003 bpp
0.001	56.22 dB	33.39 dB	0.023 bpp
0.005	59.29 dB	35.12 dB	0.043 bpp

Table 12: Fusion - 4C LIC-TCM (Depth target)

λ-config	**Average PSNR**	**Average Comprehensive Similarity**	**Average Bit-rate**
0.0001	47.17 dB	28.49 dB	0.003 bpp
0.001	55.84 dB	33.03 dB	0.024 bpp
0.005	61.19 dB	35.65 dB	0.034 bpp

Table 13: Fusion - Convolutional MM-LIC-TCM

λ-config	**Average PSNR**	**Average Comprehensive Similarity**	**Average Bit-rate**
0.0001	47.32 dB	28.40 dB	0.003 bpp
0.001	56.99 dB	33.47 dB	0.022 bpp
0.005	58.45 dB	35.01 dB	0.037 bpp

Finally, more detailed information about the architectures' time complexity, such as encoding and decoding time, GFLOPS, and the amount of parameters is shown in Table 14 and Table 15. The former table corresponds to the architectures that target RGB as output and the latter corresponds to the architectures that target depth as output. When inspecting these tables, it is evident that the 4C architectures are implemented as expected, since the performances of the 4C architectures are identical to the baselines. Furthermore, we see expected increased complexity of the fusion architectures, especially in the case of the Attention architecture.

Table 14: RGB-targeted models.

	Encoding time	**Decoding time**	**Parameters**	**GFLOPS**
Baseline - RGB	62 ms	67 ms	45.18 M	851
Fusion - Attention	76 ms (↑ 23%)	75 ms (↑ 12%)	53.53 M (↑ 18%)	1100
Fusion - 4C	63 ms	67 ms	45.18 M	850

Table 15: Depth-targeted models.

	Encoding time	**Decoding time**	**Parameters**	**GFLOPS**
Baseline - Depth	62 ms	66 ms	45.17 M	843
Fusion - Convolutional	77 ms (↑ 24%)	66 ms	47.6 M (↑ 5%)	1016
Fusion - 4C	62 ms	66 ms	45.17 M	856

4.2 Qualitative

A natural path forward after covering the quantitative results is to display the results in the form of image reconstructions and RGB-D 3D renderings. The former is a common practice in the field of image compression, while the latter is not a common sight in research papers. The decision to display 3D renderings is based on the fact that it is hard to perceive the subjective quality of a depth image, hence a creative workaround is to align the depth reconstruction frame with the ground-truth RGB frame and display it in a 3-dimensional environment. This way, it enables us to perceive the subjective quality of the depth image. The 3D renderings were done using Open3D [37].

In this section, some hand-picked reconstruction and renderings are displayed. In the depth section, we display 3D renderings with depth reconstructions, and in the RGB section, we display some RGB reconstructions.

4.2.1 Depth

In this example, a 3D rendering of Test Image 5 in the test set is used, see Figure 28. In Figure 29, the ground truth rendering is displayed. For Figure 30 (Depth baseline), 31 (Convolutional MM-LIC-TCM), the ground truth RGB is jointly used with the reconstructed depths to form 3D renderings.

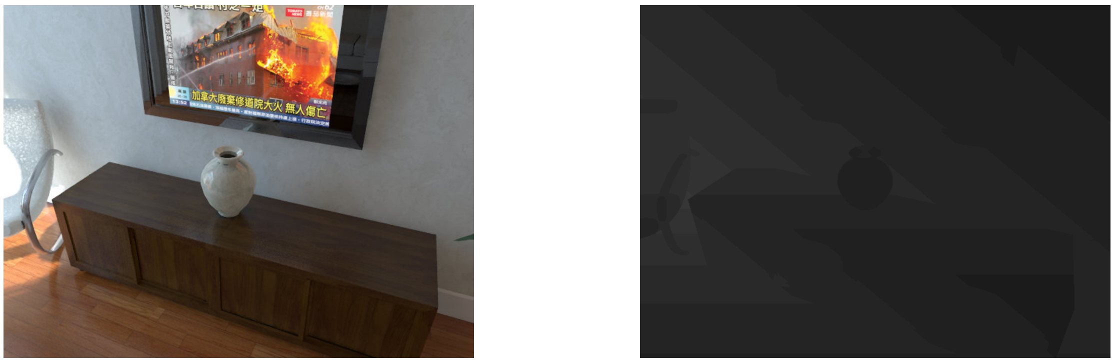


Figure 28: Test Images 5, ground truth RGB-D pair. Used in test dataset.

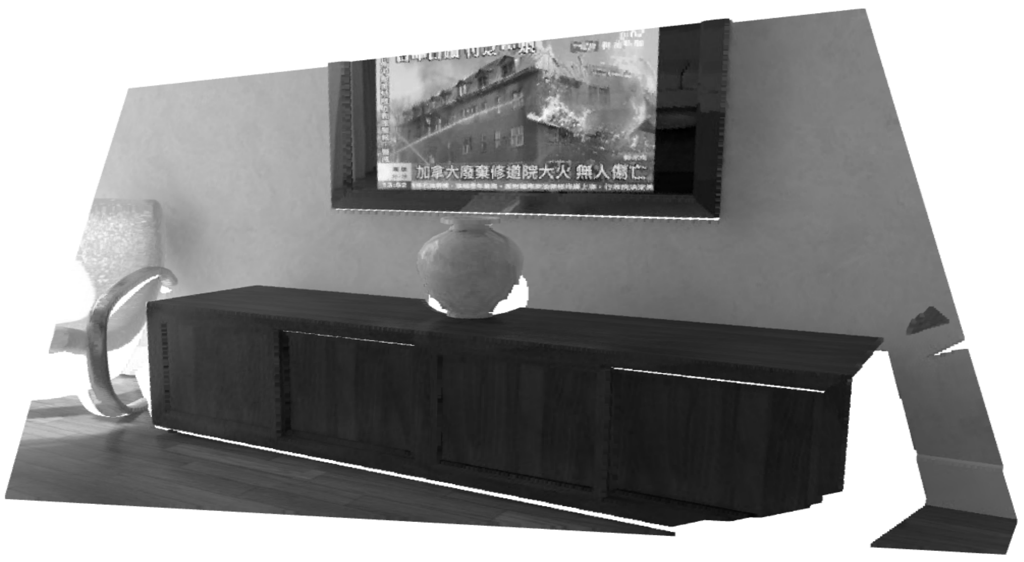


Figure 29: Rendering of the ground truth RGB-D pair.

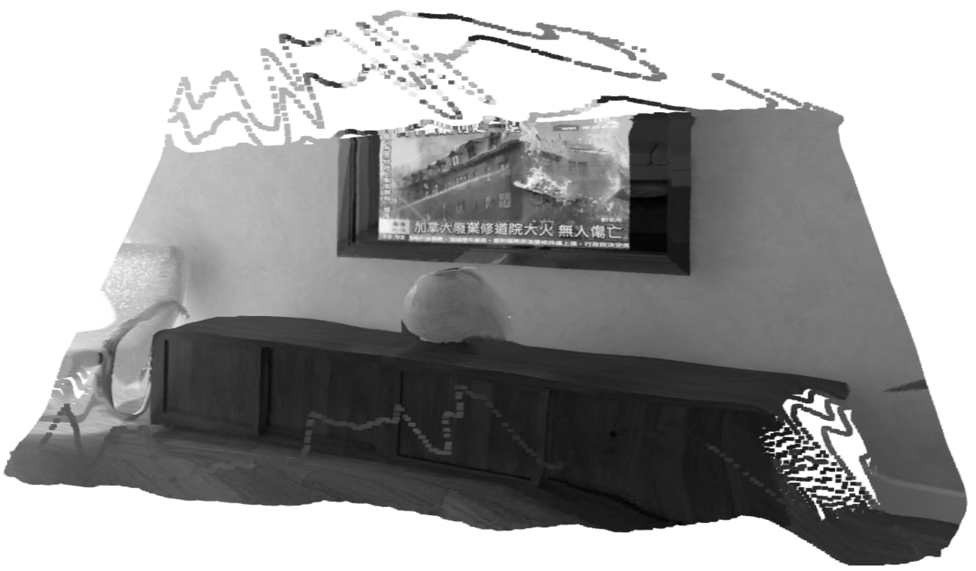


Figure 30: Rendering of reconstructed depth, using depth baseline. The following results were generated for this sample: λ: 0.001 |PSNR: 59.35 dB |0.010 bpp.

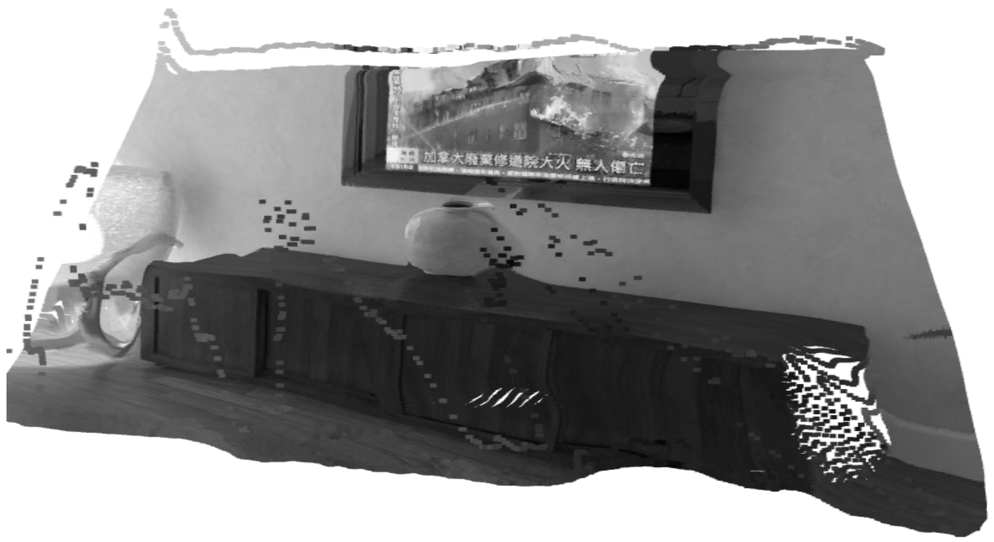


Figure 31: Rendering of reconstructed depth, using Convolutional MM-LIC-TCM architecture. The following results were generated for this sample: λ: 0.001 |PSNR: 60.47 dB |0.020 bpp.

It can be observed from the reconstructed images that the quality is not very high. Numerous "flying pixels" are visible in the results from both the baseline model and the convolutional model, particularly in the top areas. Given the minor enhancement our fusion model has achieved, it is challenging to discern any significant enhancement.

4.2.2 RGB

In this reconstruction, a sample from the SUNRGB-D [44]–[47] dataset is used. The frame is number 0000118 from KinectV2 using ground truth RGB jointly with the depth BFX as input to our models. For these reconstructions, the lambda was set to 0.0130. A comparison of the original RGB image versus some sampled regions from the different architectures can be seen in Figure 32. The individual reconstructions can be seen in Figures 33 to 35.

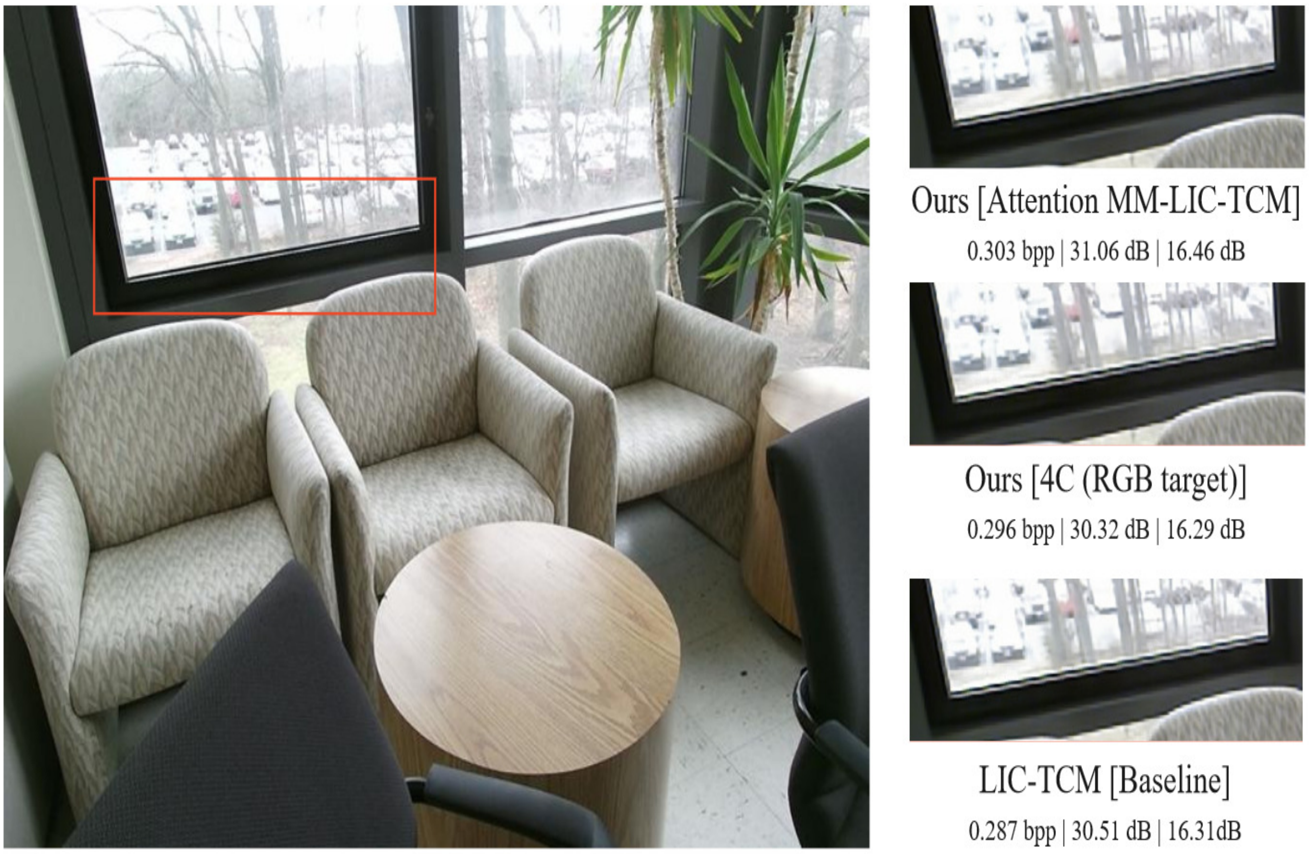

Ours [Attention MM-LIC-TCM]
0.303 bpp | 31.06 dB | 16.46 dB
Ours [4C (RGB target)]
0.296 bpp | 30.32 dB | 16.29 dB
LIC-TCM [Baseline]
0.287 bpp | 30.51 dB | 16.31dB

Figure 32: Visualization of decompressed images of frame 0000118, KinectV2, from SUNRGB-D dataset. The large figure to the left is the original image. The subfigures are titled as "Method | Bit rate | PSNR | MS-SSIM".

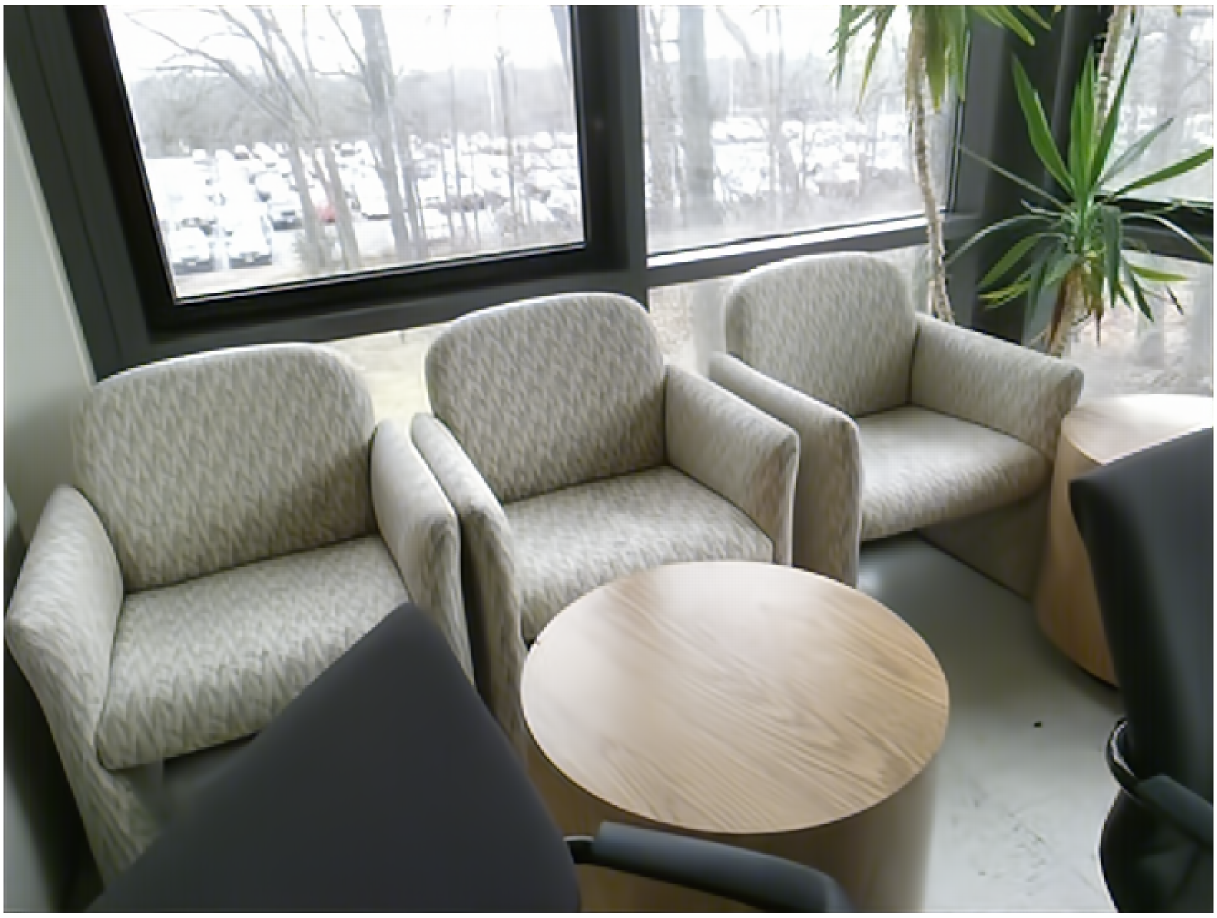


Figure 33: Visualization of decompressed images of frame 0000118, KinectV2, from SUNRGB-D dataset. Reconstructed with Attention MM-LIC-TCM. The following results were generated for this reconstruction: 0.303 bpp | PSNR: 31.06 dB | MS-SSIM: 16.46 dB.

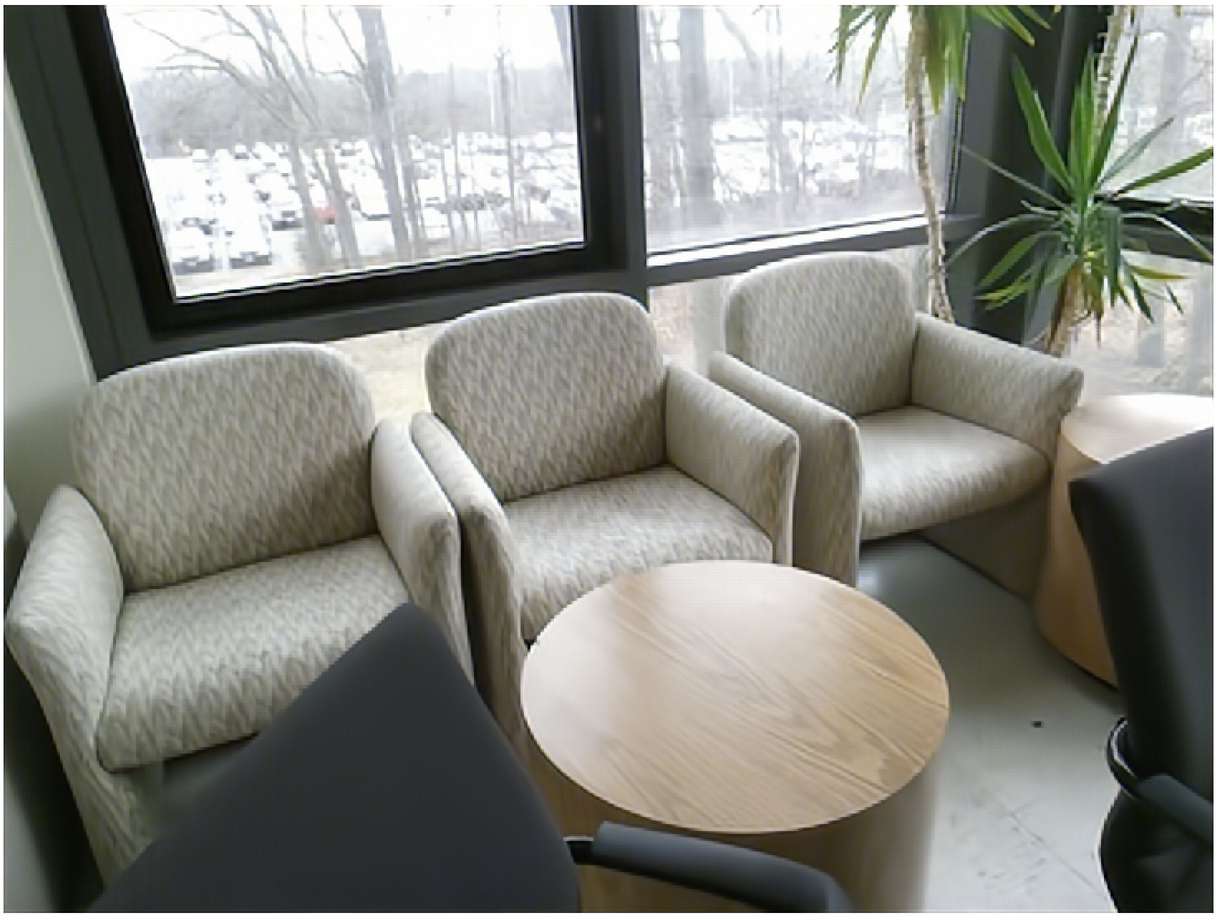


Figure 34: Visualization of decompressed images of frame 0000118, KinectV2, from SUNRGB-D dataset. Reconstructed with RGB Baseline. The following results were generated for this reconstruction: 0.287 bpp | PSNR: 30.51 dB | MS-SSIM: 16.31 dB.

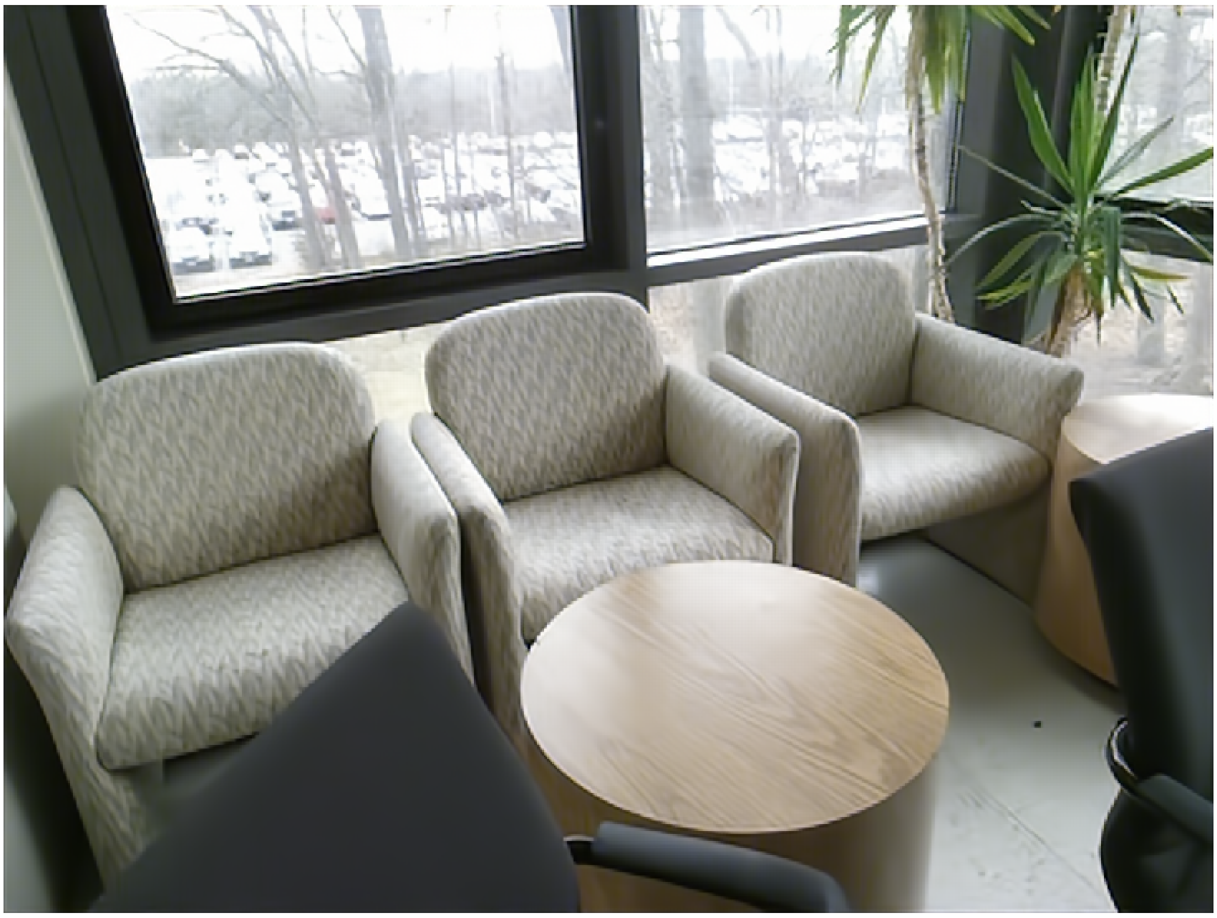


Figure 35: Visualization of decompressed images of frame 0000118, KinectV2, from SUNRGB-D dataset. Reconstructed with 4C LIC-TCM (RGB target). The following results were generated for this reconstruction: 0.296 bpp | PSNR: 30.32 dB | MS-SSIM: 16.29 dB.

5 Discussion

In this section, the quantitative and qualitative results and the implications of these are discussed. A general discussion regarding the topic of multimodal compression will take place and cover important and relevant properties such as complexity and robustness and relate them to the results. Thereafter, a detailed reasoning behind the implemented multimodal architectures will be provided.

Finally, some unmentioned and unfinished explorations will be covered, along with the reason behind them being unfinished.

5.1 Quantitative and qualitative results

While the research field of multimodal compression is small, it is evident that feasible and promising results can be produced. In the case of quantitative results, a clear and interesting improvement was found by introducing depth as a modality, elevating the foundational LIC-TCM architecture. In other words, by experimentally augmenting the LIC-TCM architecture with two new architectures, Attention MM-LIC-TCM and 4C LIC-TCM (RGB target), to include depth as a modality, an increase in PSNR was shown compared to the standard LIC-TCM architecture.

However, what is quite surprising is the strong quantitative performance of the 4C LIC-TCM (RGB target) model despite its simple construction. While the Attention MM-LIC-TCM architecture clearly introduces an increase in both architectural complexity and runtime, it does not make up for it in terms of performance when compared to 4C LIC-TCM (RGB target). But this result is not conclusive and in reality needs further investigation. Since the Attention MM-LIC-TCM architecture extends the attention mechanism it might cause a greater need for more training data to perform better. In general, transformers require a huge amount of data to be properly trained and the dataset used for this paper is in reality quite small. As an example, the authors in LIC-TCM use a quite more sophisticated dataset with triple the amount of data compared to the dataset in this paper. However, considering the long training times per architecture and the number of different configurations per architecture, it would not be feasible to have such large datasets and also deliver the results of our thesis within the assigned time limit.

Looking at the reconstructions in Section 4.2.2 it can be difficult to appreciate any major differences. We reason that these models are already very efficient, and historically, quantitative progress is done incrementally. However, in some cases, we notice that the fusion model (Attention MM-LIC-TCM) is better at smoothening out edges in images. As an example of this, see how Attention MM-LIC-TCM better reconstructs the edges of the window in Figure 32.

Turning our attention to using RGB images as a modality, we can see that the model performance is overall improved in terms of both PSNR and comprehensive similarity, regardless of which fusion strategy is used. It validates our hypothesis that the presence of an additional modality can indeed enhance compression. The improvement is particularly pronounced concerning comprehensive similarity, as expected, given that the loss is optimized based on this specifically designed metric. Additionally, the enhancement becomes more apparent as the bitrate increases, as observed in Figure 27. Surprisingly, the 4C LIC-TCM (Depth target) model significantly outperforms the others at a high bitrate while it drops greatly in the middle, making it the lowest among the three models. Thus, we sense that the training of the network can still be optimized. Fine-tuning the configuration in future work could provide further insights.

Compared to the reconstruction of RGB images, the reconstructed depth images exhibit relatively lower quality. The compression of depth images requires higher precision. For example, in RGB images, a deviation in pixel intensity might be negligible to human eyes. However, in depth images, such a deviation can result in a pixel becoming a "flying pixel", which is very noticeable and easily detected. Once again, the difference between the baseline model and our fusion models is minor, but it is noticeable that there are fewer outliers for the Convolutional MM-LIC-TCM model, particularly at the top of the reconstructed image.

An overall important trade-off to consider in terms of the fusion models is quality versus performance- and time complexity. Reflecting back to Tables 14 and 15, it is evident that the Convolutional MM-LIC-TCM and Attention MM-LIC-TCM architectures introduces complexity. For Convolutional MM-LIC-TCM, the encoding time increases by 24% compared to the corresponding baseline model, while the decoding time remains unaffected. Thus, the overall time complexity is raised by 12%. Likewise, the number of parameters is also a bit higher, which reaches 47.6M (5% more than the baseline model). For Attention MM-LIC-TCM, we see an overall increased time complexity of 17% as well as a significant increase of 18% in number of parameters. Considering this added complexity, is it worth a 0.17 increase in PSNR (see Figure 25 for 0.0130 lambda)? We therefore repeat the need for deeper investigation of Attention MM-LIC-TCM for more conclusive results of the architecture's true capabilities, especially in the case of Attention MM-LIC-TCM with lambda set to 0.05. With all this said, it makes the choice of 4C LIC-TCM (RGB target) architectures appealing because of its low implementation cost and simultaneously being unscathed from an increase in amount of parameters and increase in time complexity.

To conclude this subsection, a final word of caution shall once again be echoed as previously stated in the thesis. While PSNR and MS-SSIM are good perceptual metrics for the field of image compression, it is *not* an objective perceptual metric. Additionally, it may also be difficult to appreciate the small margins in the quantitative results when the reconstructions/renderings are still seemingly similar, but progress in learned lossy image compression started small yet today supersedes standard image codecs.

5.2 Our implementations

As a starting point, a decision was made to prioritize extending already state-of-the-art models rather than constructing an architecture from scratch. Not only to limit the workload considering the time constraints within the thesis, but rather the fact that if an architecture not intended for multimodality can be integrated with multimodal capabilities with low implementation cost and elevated results, then the implications of this is that multimodality can be seen as a natural option to investigate for optimization.

The motivation behind the different multimodal architectures was to have maximum reach in the different possible strategies. In our implementations we have three different strategies: model-agnostic fusion (4C LIC-TCM), model-based fusion by using attention mechanism (Attention MM-LIC-TCM) and convolution (Convolutional MM-LIC-TCM). This way, we get a wider reach in our results and insights.

The reason to let one architecture focus on RGB as a target output or the other focus on depth as a target output was based on the papers the architectures originated from. However, we do not exclude the possibility of just arbitrarily changing the input modalities and see how well it performs. Considering the flexibility of the given implementations, the order of modalities could be changed or even additional modalities could be added and experimented with (for example a segmentation mask). Our fusion architectures are highly flexible and adaptable to most learned image compression architectures, so even other learned image compression architectures could be used as a foundational model instead of LIC-TCM. To state it clearly, we believe that our implementations are generalizable.

The reasoning behind only making changes to the encoder network (g_a) was mainly twofold. Firstly, one of the objectives was to keep the complexity low and secondly, in [4], an ablation study was done removing p_a and l_a showing that their model still performed well. We decided that we would do the same for Attention MM-LIC-TCM, but for consistency, we decided to only affect the encoder network for both fusion models. But of course, it would be interesting to expand on the decoder network (g_s) and run new training and evaluations.

Another important aspect is the distortion term which calls for dedicated attention. The distortion term is of *critical* importance in terms of reconstruction quality, even for high lambdas (although high lambda but poor distortion function will perform poorly). In the case of depth compression, it was clear early in the thesis that MSE or MS-SSIM as a distortion calculation was not enough to generate satisfactory reconstructions. Hence, comprehensive similarity but adjusted for MS-SSIM instead of SSIM was our novel adjustment to improve the results.

Conversely, a better distortion calculation than MSE should probably have been used for the RGB reconstruction-based architectures. While a simple, and often a good-enough metric, there is definitely room for more sophisticated distortions terms, even MS-SSIM.

5.3 Machine learning-based data compression

In this thesis, we have focused on machine learning-based data compression approaches. However, there are also conventional compression tools, such as JPEG. The main reason we do not compare our models with these traditional methods is that we believe our models have not yet reached their optimal performance. Specifically, the dataset we used consists of around 120k RGB-D pairs, which is far from sufficient. Thus, comparing our models with conventional methods might not be a fair comparison. Therefore, we investigate the *relative* improvement of our models compared to the baseline. Nevertheless, it is reasonable to argue that since a well-trained LIC-TCM outperforms the conventional tools [5], our models can reach a state where it outperforms conventional compression methods as well.

It can be seen that any ML-based methods require considerable data to achieve satisfying performance and also the training can sometimes be very time-consuming. This is obviously a significant limitation. Moreover, the ML-based models tend to have a relatively weaker universality. During our work, we find that the pre-trained LIC-TCM model is unable to compress depth images well, although it works nicely with RGB images. Conversely, JPEG, as an example of the traditional methods, can handle different kinds of image data. Furthermore, great consideration has to be taken into account for what is the application of the machine learning-based compression model. In our case, the models are trained on indoor scenes and if one was to use the model in an outdoor environment, then one could expect serious degradation of quality - but this aspect was not explored in any detail. Finally, one last limitation is the amount of parameters that has to be saved along with the model. Depending on chosen model size, it can vary from 500 MB for a small model to 900 MB for a large model in the case of LIC-TCM.

To conclude, ML-based models undeniably have many advantages, but their limitations also need to be considered, such as the requirements for data volume and generalizability mentioned above.

5.4 Other explored areas

During the course of the thesis, some areas are explored in the search for innovative solutions. One of these was based on a whitepaper from Intel [48]. The idea of the paper is to apply a colormap to some depth image according to some specific color mapping scheme, and then being able to revert the process by applying an inverse color mapping scheme.

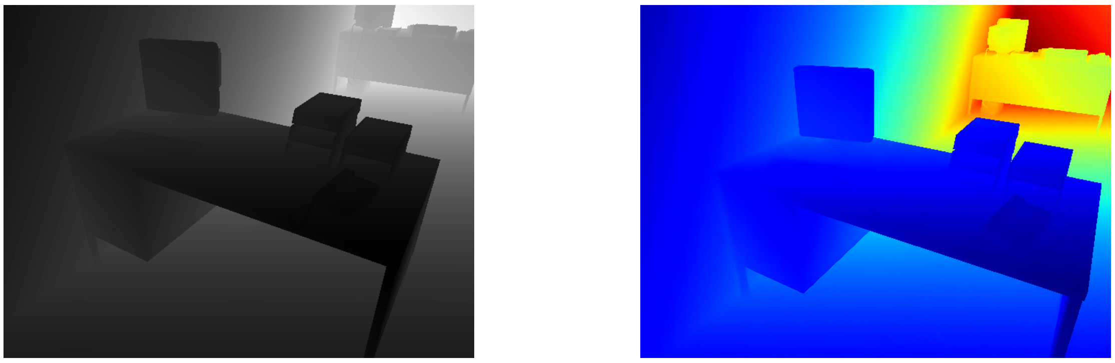


Figure 36: Left: Original depth image (but for visualization, higher contrast). Right: Jet colormapping applied to the depth image.

We were interested in investigating whether it is feasible to utilize a standard RGB compression model for compressing depth data. If a standard RGB compression model could be used for depth compression, then one could potentially create one universal model with one set of weights for RGB compression and one set of weights for depth compression. The intended strategy was to apply either some Jet or Turbo [49] color mapping, and then pass it through standard LIC-TCM. See Figure 36 for the original depth image applied with Jet colormap. To reconstruct the depth image, the LIC-TCM RGB reconstruction would be sampled according to some inversed color mapping scheme to retrieve the depth.

However, the cause for discontinuing this strategy was simply related to the sampling schemes. Firstly, the process of accurately reverting the color-mapped depth image back to its grayscale form is not trivial. Secondly, sampling a 16-bit depth image to a 24-bit RGB image will introduce a loss of information if one wants to stay true to the relevant color mapping spectrum. In summary, the color mapping strategy proved to be a significant time sink, diverting attention away from the more critical aspects of this thesis. Consequently, we made the decision to discontinue this line of work.

6 Conclusions and Outlook

In this thesis, we propose three different unique architectures that extend a single-modality lossy learned image compression architecture to encompassing depth as an additional input modality. Based on our results, it appears that including depth as a modality offers the possibility to elevate single-modal RGB architectures, sometimes without any additional complexity cost. Conversely, adding RGB as a modality to a depth lossy learned compression architecture implies the same results. Furthermore, we believe that our implementations are generalizable and can be experimented with and adapted to most learned image compression architectures.

However, these implementations and strategies can still be optimized and warrant further investigation. To achieve more conclusive results, much more investigation should be done. Most importantly, we suggest enforcing much longer training sessions with much larger and much more sophisticated dataset(s). For example, the Habitat-Matterport 3D Research Dataset[50] by Meta is a good example of what kind of dataset could be used.

Another interesting option is to investigate the distortion term further. Unfortunately, the option of exploring RGB reconstruction models with MS-SSIM as a distortion term was not possible due to time constraints. Furthermore, we believe that it should not be stopped at MS-SSIM. Possibly a more hand-crafted distortion term could be designed to combine multiple modalities. Not strictly related to multimodality, but as an example, researchers at Google (same authors as the hyperprior paper) propose a new perceptual quality metric that should be competitive with MS-SSIM in [51].

References

[1] I. Bousquette, "Data deluge: Businesses struggle with tmi," *WSJ*, 2024, Available: `https://www.wsj.com/articles/data-deluge-businesses-struggle-with-tmi-5e41cca1`, visited 2024-02-10.

[2] N. S. Suhasini and S. Puli, "Big data analytics in cloud computing," in *2021 Sixth International Conference on Image Information Processing (ICIIP)*, IEEE, vol. 6, 2021, pp. 320–325.

[3] M. Chen, P. Zhang, Z. Chen, Y. Zhang, X. Wang, and S. Kwong, "End-to-end depth map compression framework via rgb-to-depth structure priors learning," in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, pp. 3206–3210. DOI: `10.1109/ICIP46576.2022.9898073`.

[4] A. Gnutti, S. D. Fiore, M. Savardi, Y.-H. Chen, R. Leonardi, and W.-H. Peng, *Lidar depth map guided image compression model*, 2024. arXiv: `2401.06517 [eess.IV]`.

[5] J. Liu, H. Sun, and J. Katto, *Learned image compression with mixed transformer-cnn architectures*, 2023. arXiv: `2303.14978 [eess.IV]`.

[6] J. Zou, Y. Han, and S.-S. So, "Overview of artificial neural networks," in *Artificial Neural Networks: Methods and Applications*, D. J. Livingstone, Ed. Totowa, NJ: Humana Press, 2009, pp. 14–22, ISBN: 978-1-60327-101-1. DOI: `10.1007/978-1-60327-101-1_2`. [Online]. Available: `https://doi.org/10.1007/978-1-60327-101-1_2`.

[7] M. Ohlsson and P.Edén, *Lecture notes in introduction to artificial neural networks and deep learning (extq40, lth, lunds university)*, 2022.

[8] M. Mandal, *Introduction to convolutional neural networks (cnn)*. [Online]. Available: `https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/` (visited on 03/25/2024).

[9] D. W. Abueidda, S. Koric, R. A. Al-Rub, C. M. Parrott, K. A. James, and N. A. Sobh, "A deep learning energy method for hyperelasticity and viscoelasticity," *European Journal of Mechanics - A/Solids*, vol. 95, p. 104 639, Sep. 2022, ISSN: 0997-7538. DOI: `10.1016/j.euromechsol.2022.104639`. [Online]. Available: `http://dx.doi.org/10.1016/j.euromechsol.2022.104639`.

[10] K. O'shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: `1511.08458 [cs.NE]`.

[11] J. Liu, S. Di, K. Zhao, *et al.*, *Exploring autoencoder-based error-bounded compression for scientific data*, 2023. arXiv: `2105.11730 [cs.LG]`.

[12] U. Michelucci, *An introduction to autoencoders*, 2022. arXiv: `2201.03898 [cs.LG]`.

[13] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: `1706.03762 [cs.CL]`.

[14] Y. Bai, J. Mei, A. L. Yuille, and C. Xie, "Are transformers more robust than cnns?" *Advances in neural information processing systems*, vol. 34, pp. 26 831–26 843, 2021. arXiv: `2111.05464 [cs.CV]`.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. arXiv: `2010.11929 [cs.CV]`.

[16] Z. Liu, Y. Lin, Y. Cao, *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. arXiv: `2103.14030 [cs.CV]`.

[17] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: `1312.6114 [stat.ML]`.

[18] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019, ISSN: 1935-8245. DOI: `10.1561/2200000056`. [Online]. Available: `http://dx.doi.org/10.1561/2200000056`.

[19] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, *Neural discrete representation learning*, 2018. arXiv: `1711.00937 [cs.LG]`.

[20] A. Razavi, A. van den Oord, and O. Vinyals, *Generating diverse high-fidelity images with vq-vae-2*, 2019. arXiv: `1906.00446 [cs.LG]`.

[21] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, *Variational image compression with a scale hyperprior*, 2018. arXiv: `1802.01436 [eess.IV]`.

[22] J. Ballé, V. Laparra, and E. P. Simoncelli, *End-to-end optimized image compression*, 2017. arXiv: `1611.01704 [cs.CV]`.

[23] J. Ballé, P. A. Chou, D. Minnen, *et al.*, *Nonlinear transform coding*, 2020. arXiv: `2007.03034 [cs.IT]`.

[24] D. Minnen, J. Ballé, and G. Toderici, *Joint autoregressive and hierarchical priors for learned image compression*, 2018. arXiv: `1809.02736 [cs.CV]`.

[25] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: `10.1002/j.1538-7305.1948.tb01338.x`.

[26] J. Duda, *Asymmetric numeral systems: Entropy coding combining speed of huffman coding with compression rate of arithmetic coding*, 2014. arXiv: `1311.2540 [cs.IT]`.

[27] L. Parcalabescu, N. Trost, and A. Frank, *What is multimodality?* 2021. arXiv: `2103.06304 [cs.AI]`.

[28] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423–443, 2019. DOI: `10.1109/TPAMI.2018.2798607`.

[29] E. Kodak, *Kodak lossless true color image suite*, 1993.

[30] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, *Compressai: A pytorch library and evaluation platform for end-to-end compression research*, 2020. arXiv: 2011.03029 [cs.CV].

[31] J. Ballé, V. Laparra, and E. P. Simoncelli, *Density modeling of images using a generalized normalization transformation*, 2016. arXiv: 1511.06281 [cs.LG].

[32] J. Ballé, *Efficient nonlinear transforms for lossy image compression*, 2018. arXiv: 1802.00847 [eess.IV].

[33] D. Minnen and S. Singh, *Channel-wise autoregressive entropy models for learned image compression*, 2020. arXiv: 2007.08739 [eess.IV].

[34] C.-H. Kao, Y.-C. Weng, Y.-H. Chen, W.-C. Chiu, and W.-H. Peng, *Transformer-based variable-rate image compression with region-of-interest control*, 2023. arXiv: 2305.10807 [eess.IV].

[35] M. Jia, L. Tang, B.-C. Chen, *et al.*, *Visual prompt tuning*, 2022. arXiv: 2203.12119 [cs.CV].

[36] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 143–152. DOI: 10.1109/ICCV.2017.25.

[37] Q.-Y. Zhou, J. Park, and V. Koltun, *Open3D: A modern library for 3D data processing*, 2018. arXiv: 1801.09847 [cs.CV].

[38] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1524–1531. DOI: 10.1109/ICRA.2014.6907054.

[39] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5556–5565. DOI: 10.1109/CVPR.2015.7299195.

[40] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, 2003, 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.

[41] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[42] I. Alhashim and P. Wonka, *High quality monocular depth estimation via transfer learning*, 2018. arXiv: 1812.11941 [cs.CV].

[43] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[44] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 567–576. DOI: 10.1109/CVPR.2015.7298655.

[45] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760, ISBN: 978-3-642-33715-4.

[46] A. Janoch, S. Karayev, Y. Jia, *et al.*, “A category-level 3-d object dataset: Putting the kinect to work,” in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 1168–1174. DOI: 10.1109/ICCVW.2011.6130382.

[47] J. Xiao, A. Owens, and A. Torralba, “Sun3d: A database of big spaces reconstructed using sfm and object labels,” in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1625–1632. DOI: 10.1109/ICCV.2013.458.

[48] T. Sonoda and A. Grunnet-Jepsen, *Depth image compression by colorization for intel® realsense™ depth cameras.* [Online]. Available: https://dev.intelrealsense.com/docs/depth-image-compression-by-colorization-for-intel-realsense-depth-cameras (visited on 02/16/2024).

[49] A. Mikhailov, *Blog post: Turbo, an improved rainbow colormap for visualization.* [Online]. Available: https://research.google/blog/turbo-an-improved-rainbow-colormap-for-visualization/ (visited on 02/20/2024).

[50] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, *et al.*, “Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. [Online]. Available: https://arxiv.org/abs/2109.08238.

[51] S. Bhardwaj, I. Fischer, J. Ballé, and T. Chinen, *An unsupervised information-theoretic perceptual quality metric*, 2021. arXiv: 2006.06752 [cs.CV].

A Test Set Images

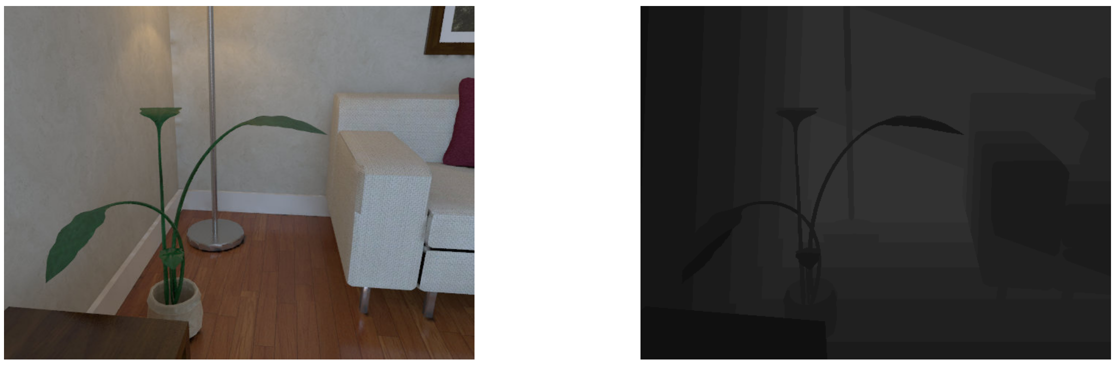


Figure 37: Test Images 1.

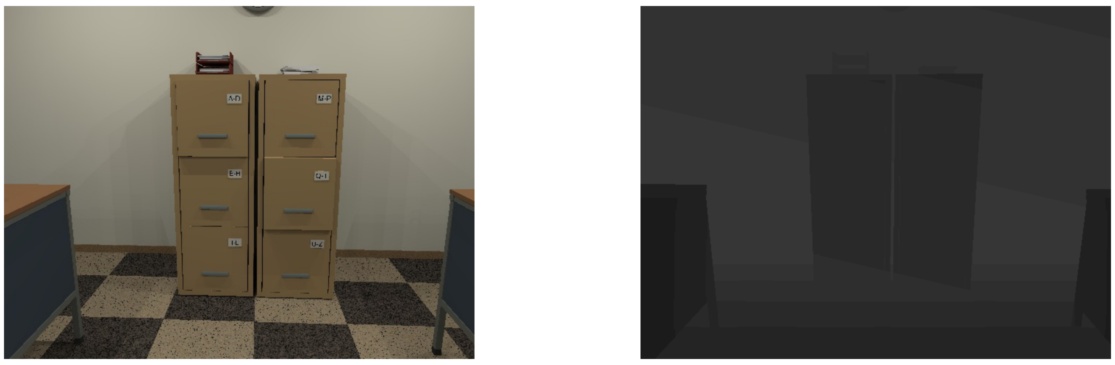


Figure 38: Test Images 2.

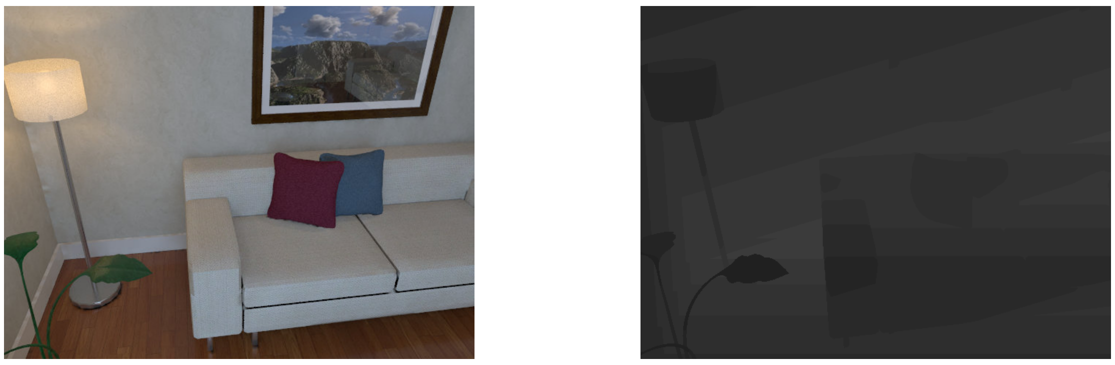


Figure 39: Test Images 3.

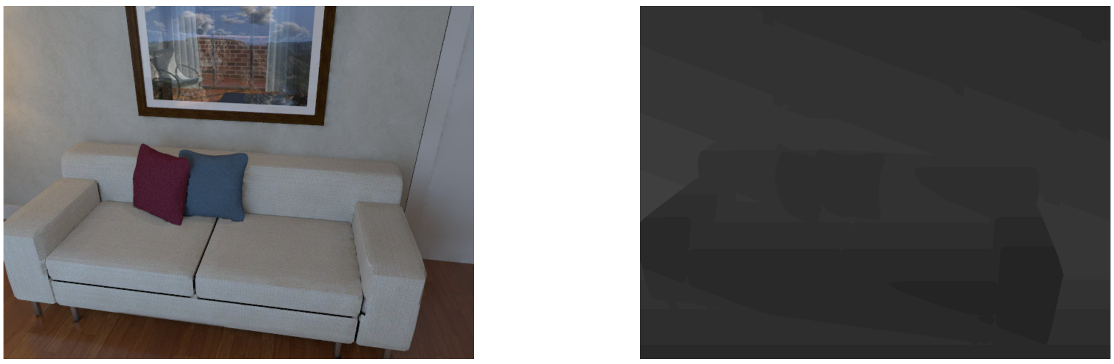


Figure 40: Test Images 4.

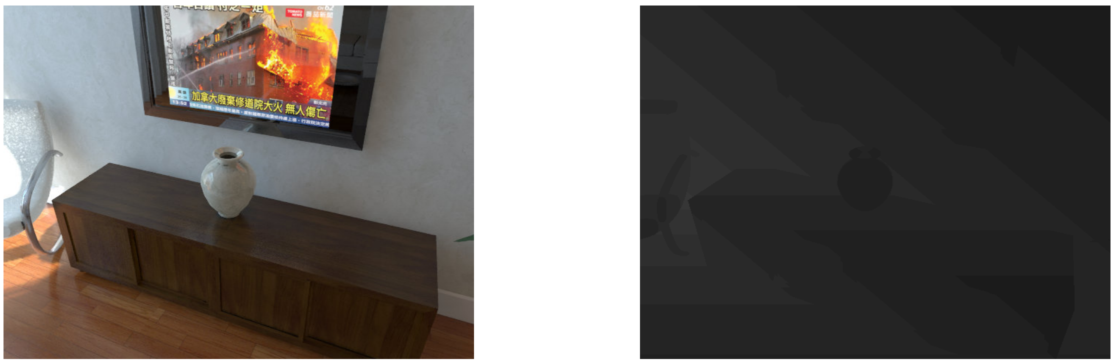


Figure 41: Test Images 5.

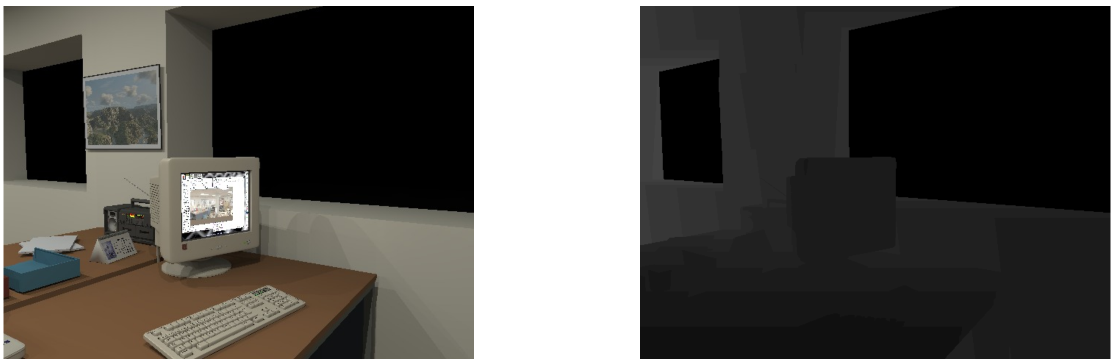


Figure 42: Test Images 6.

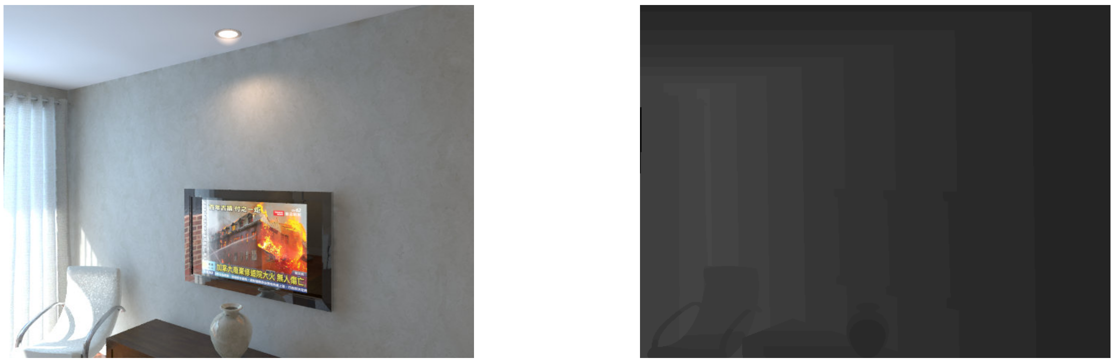


Figure 43: Test Images 7.

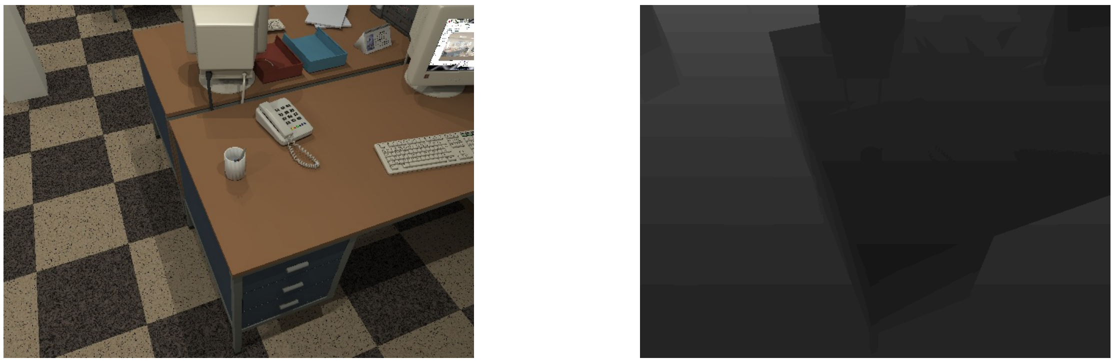


Figure 44: Test Images 8.

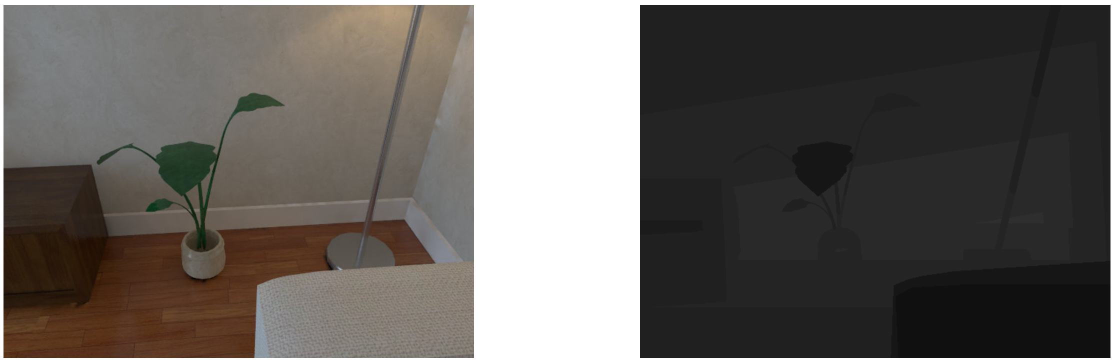


Figure 45: Test Images 9.

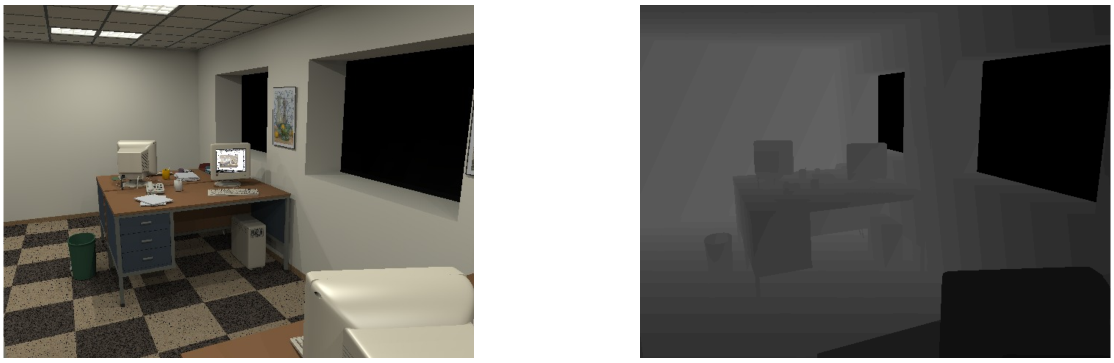


Figure 46: Test Images 10.

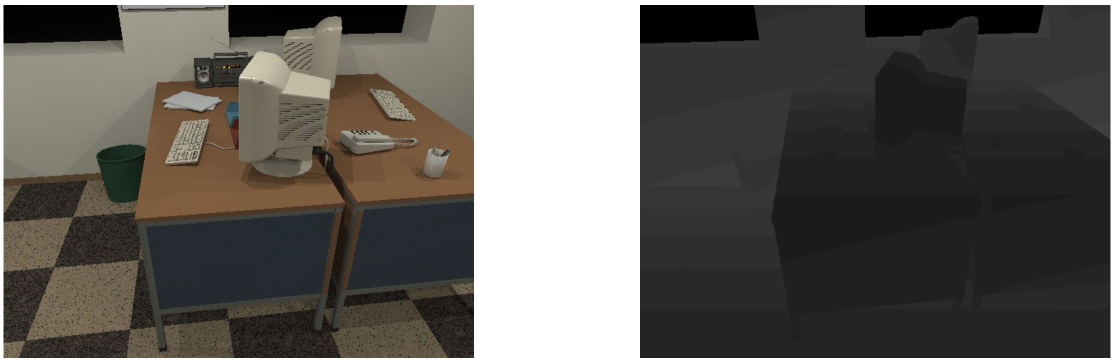


Figure 47: Test Images 11.

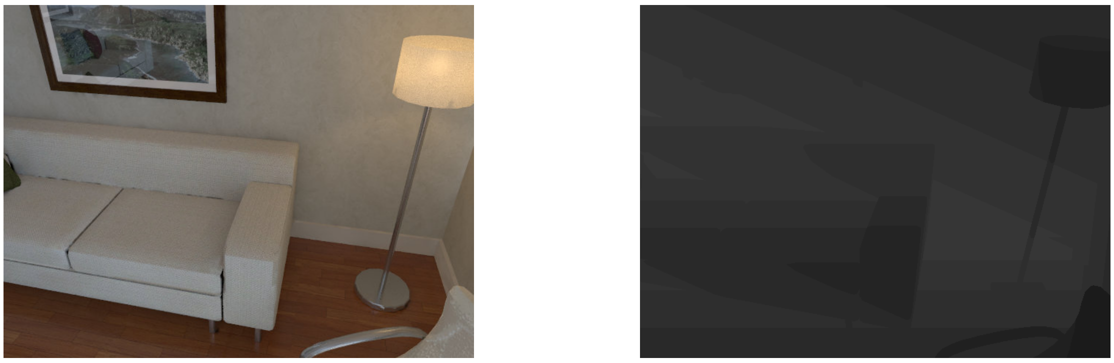


Figure 48: Test Images 12.

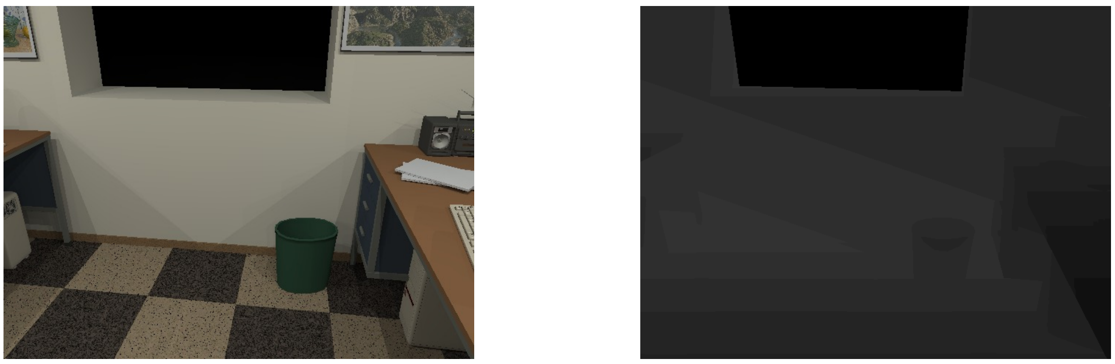


Figure 49: Test Images 13.

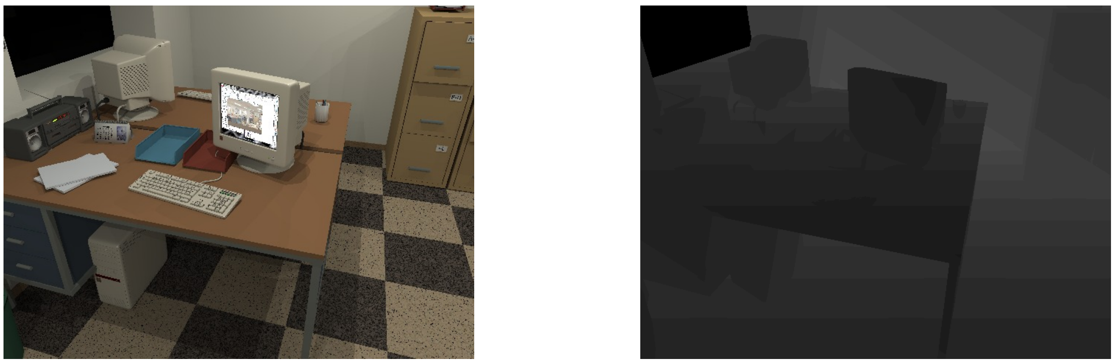


Figure 50: Test Images 14.

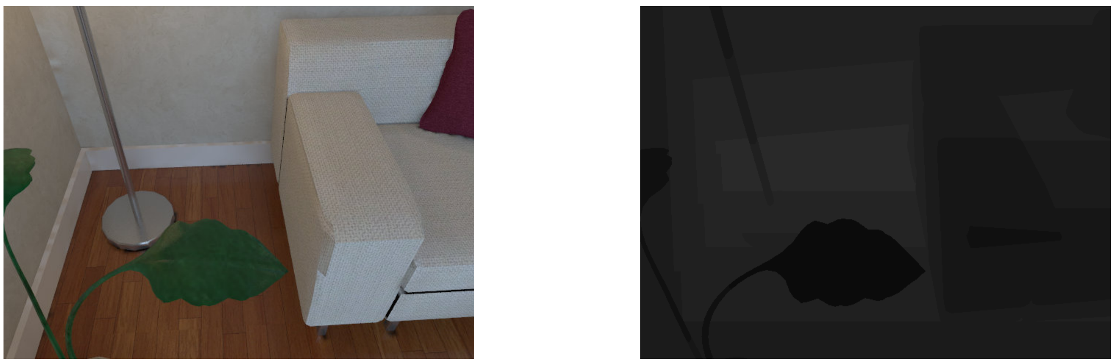


Figure 51: Test Images 15.

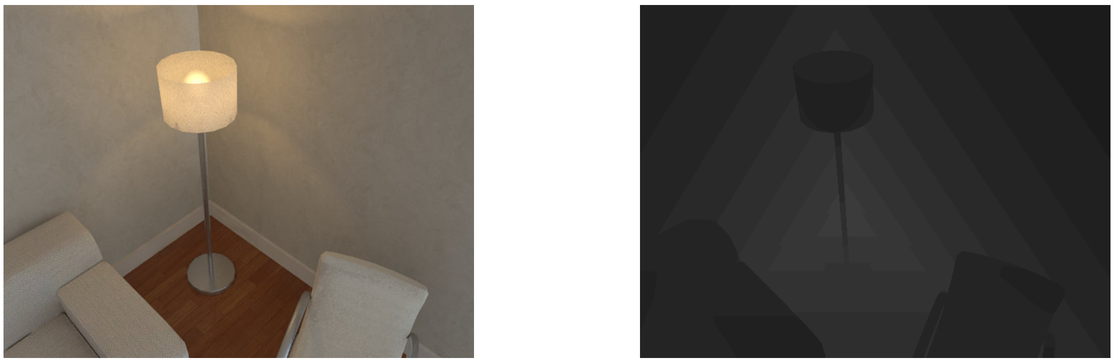


Figure 52: Test Images 16.

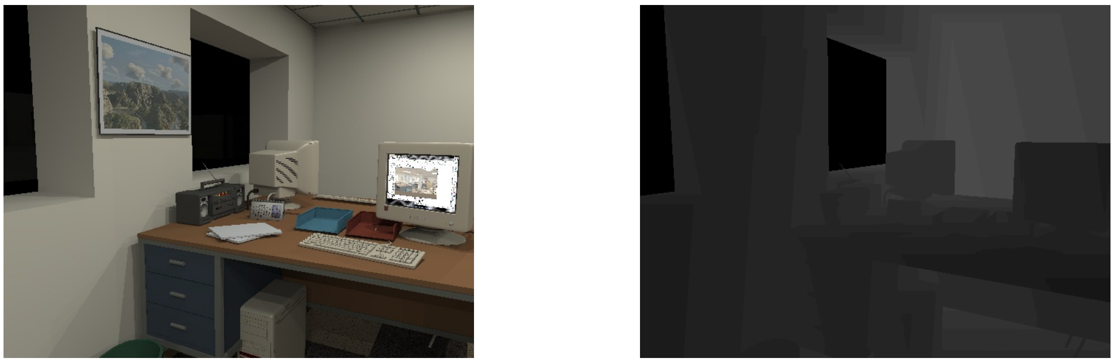


Figure 53: Test Images 17.

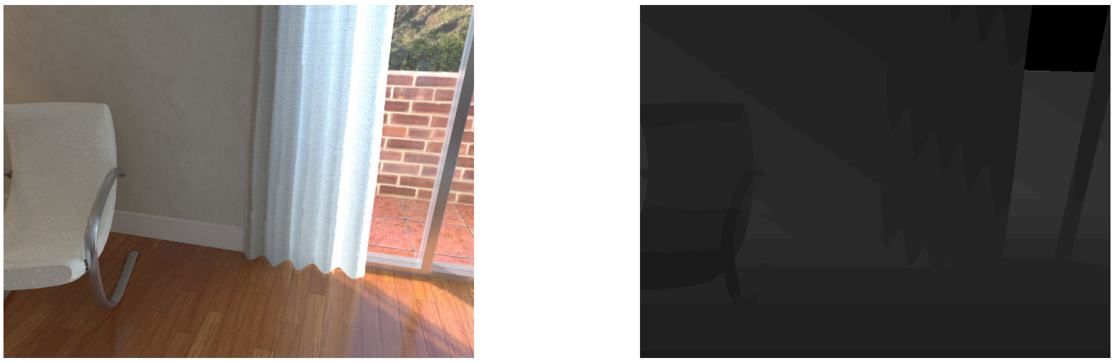


Figure 54: Test Images 18.

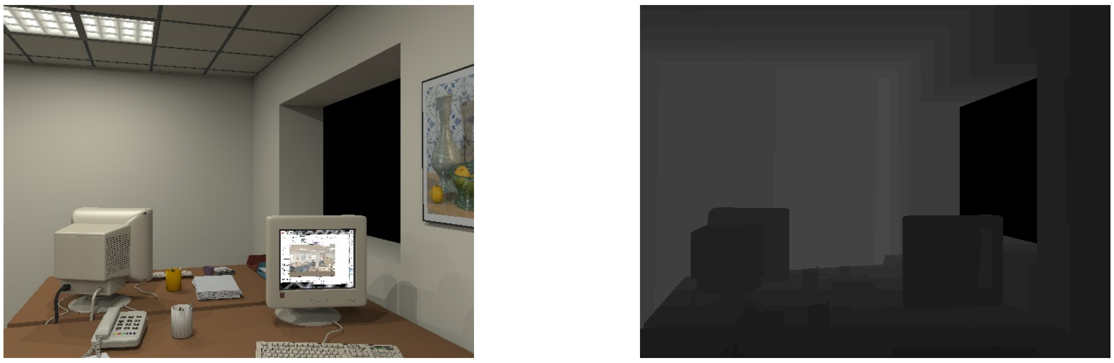


Figure 55: Test Images 19.

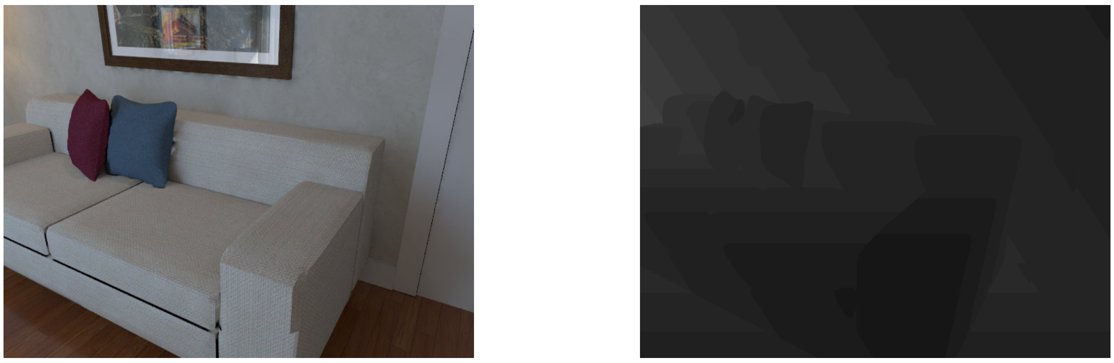


Figure 56: Test Images 20.

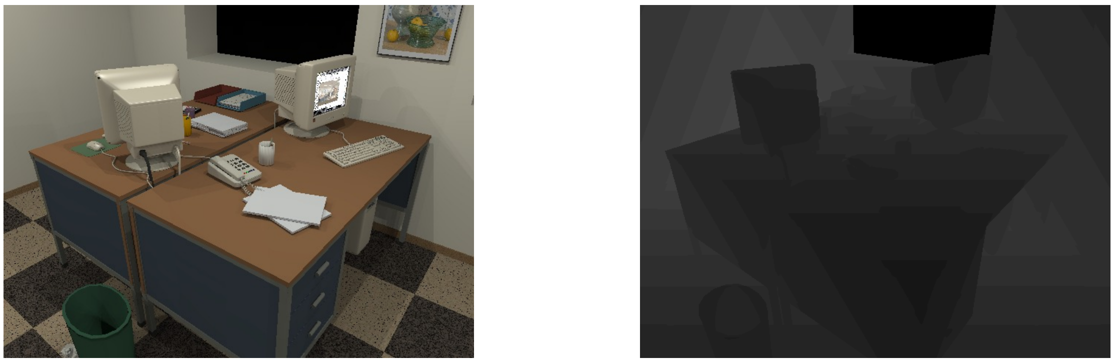


Figure 57: Test Images 21.

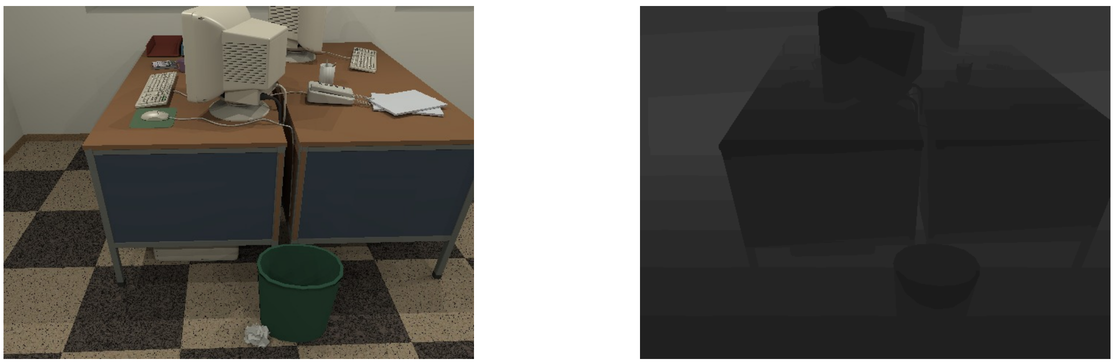


Figure 58: Test Images 22.

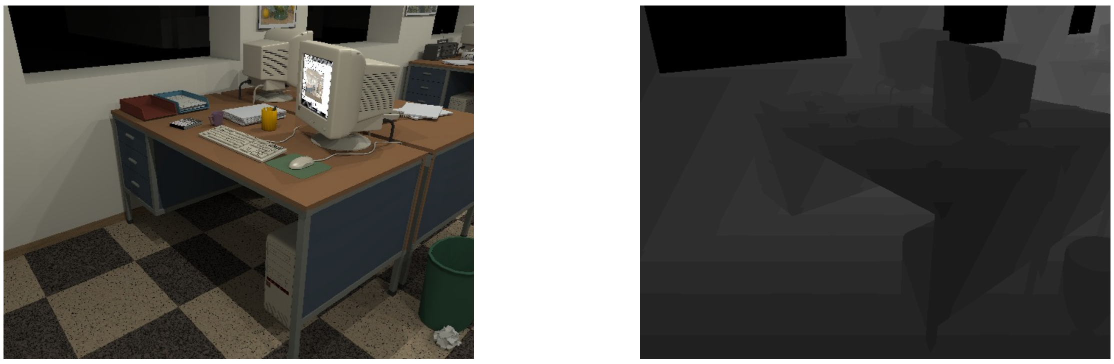


Figure 59: Test Images 23.

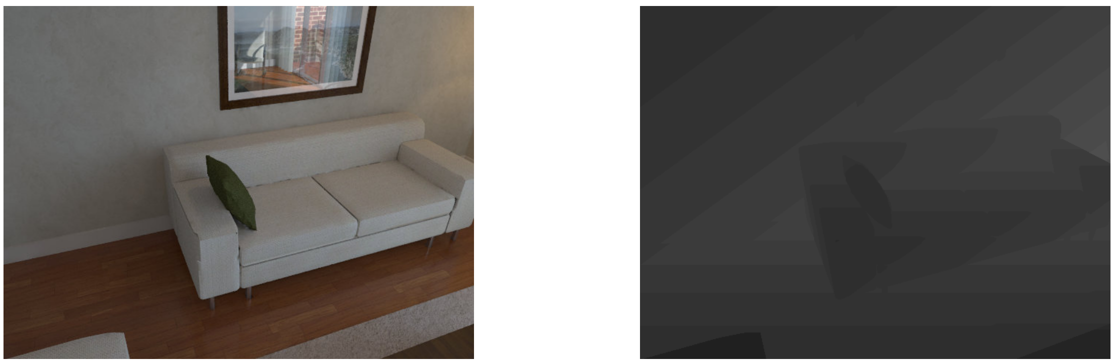


Figure 60: Test Images 24.

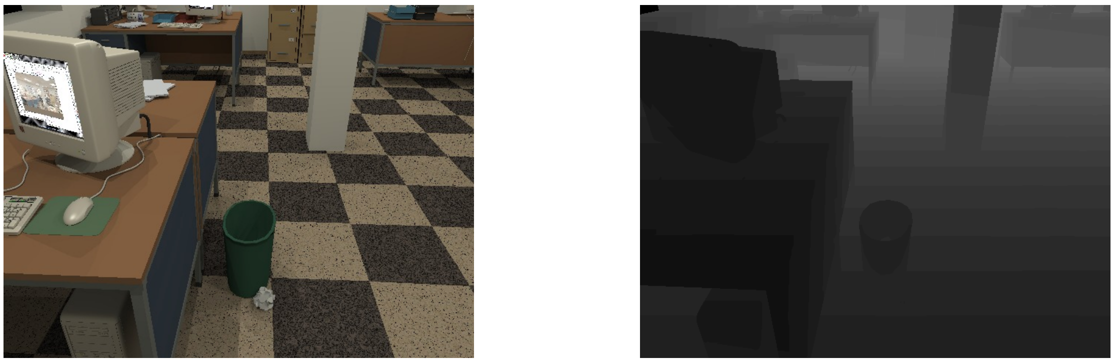


Figure 61: Test Images 25.

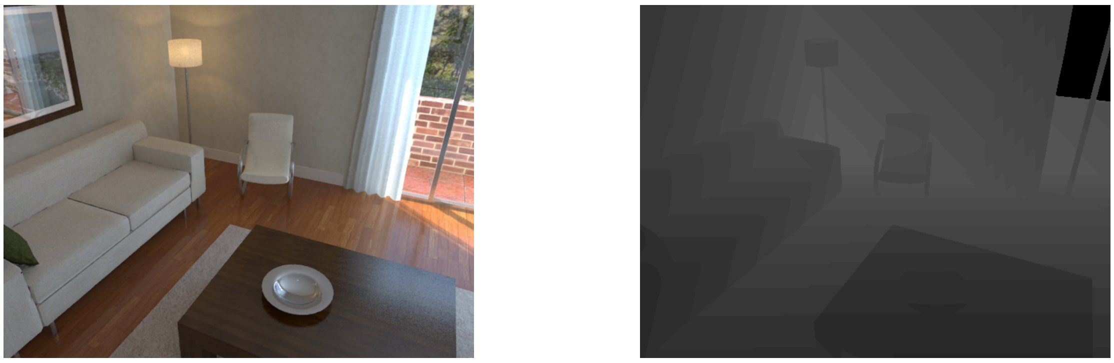


Figure 62: Test Images 26.

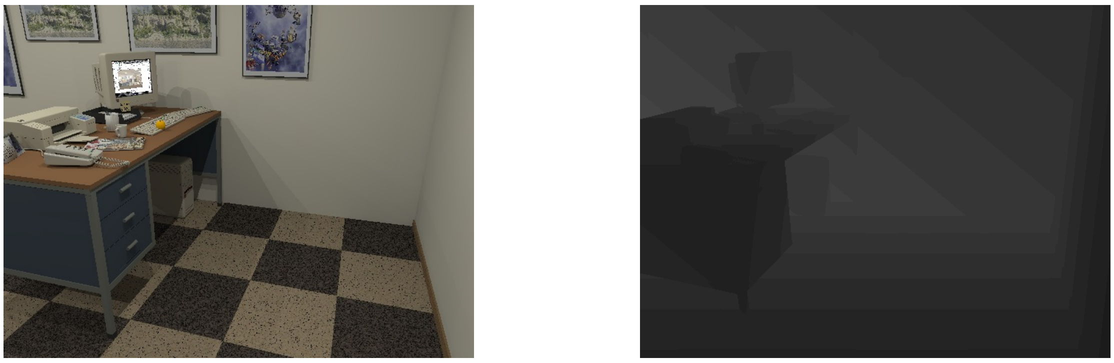


Figure 63: Test Images 27.

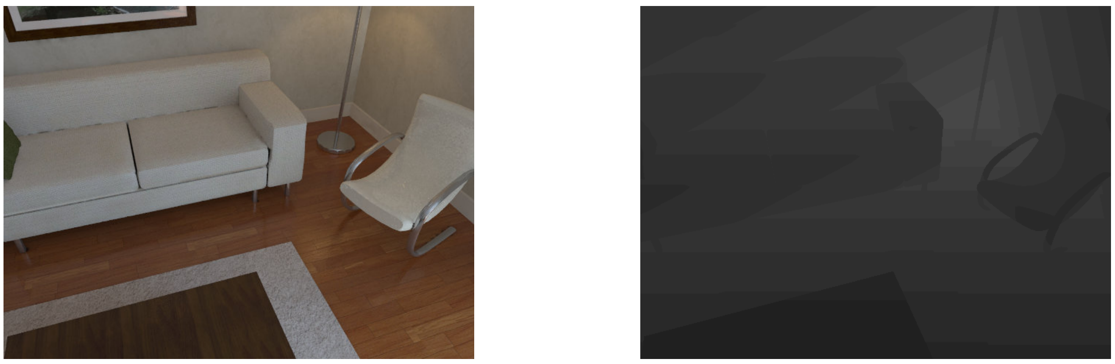


Figure 64: Test Images 28.

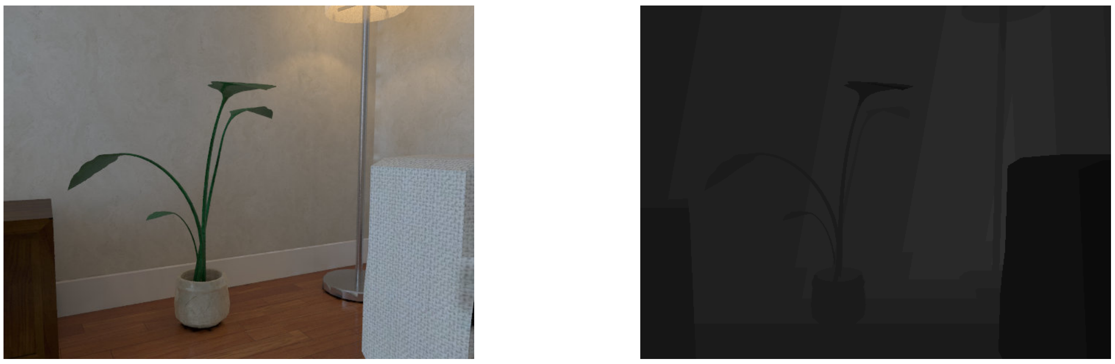


Figure 65: Test Images 29.

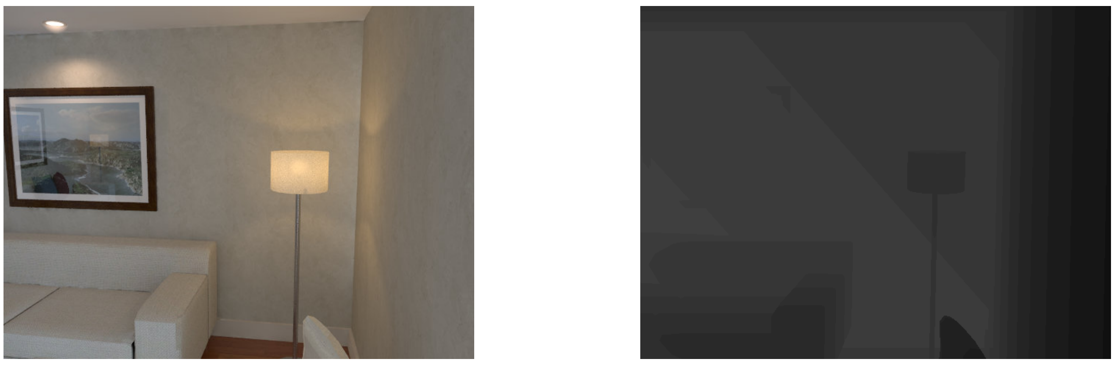


Figure 66: Test Images 30.

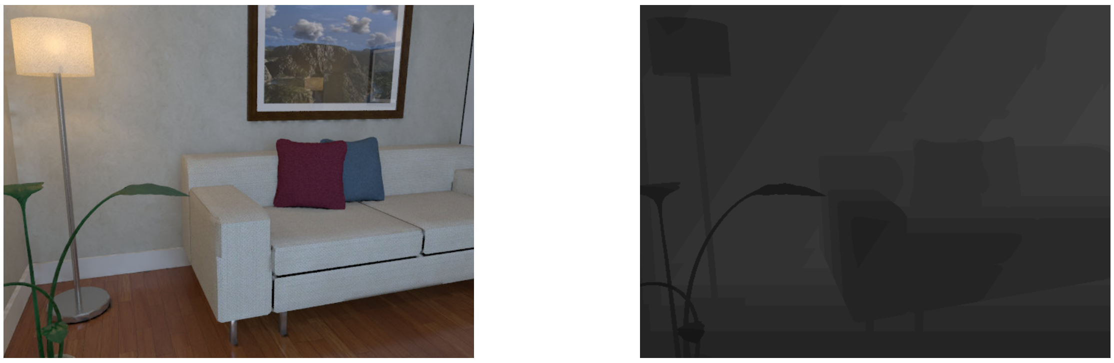


Figure 67: Test Images 31.

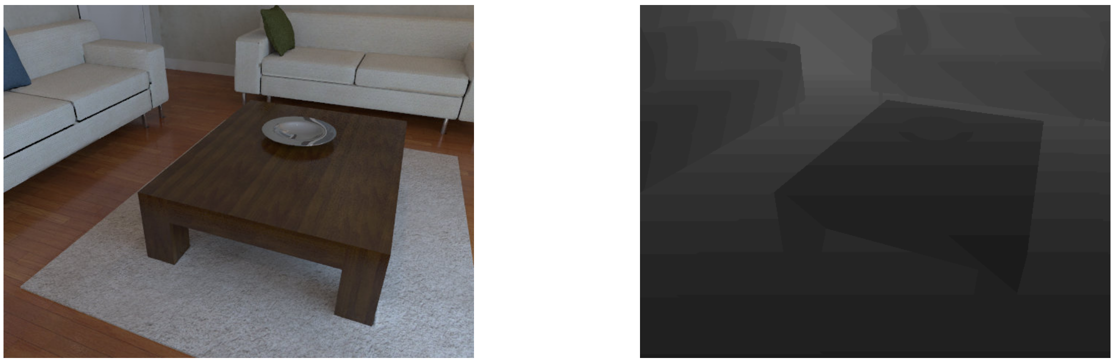


Figure 68: Test Images 32.

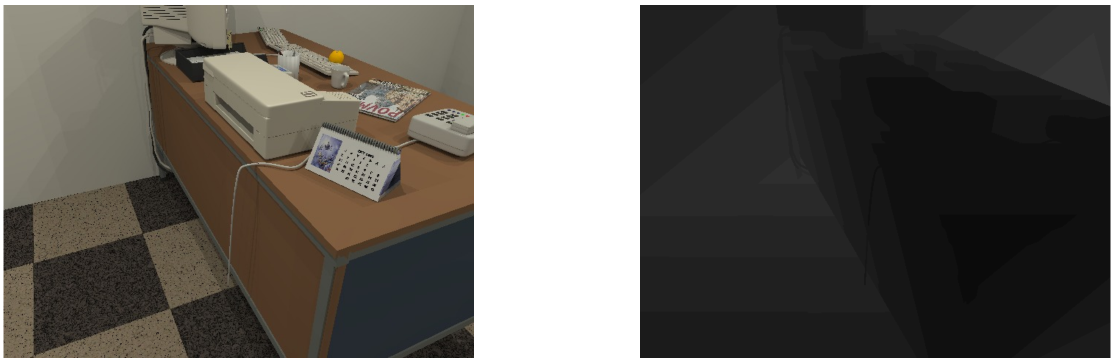


Figure 69: Test Images 33.

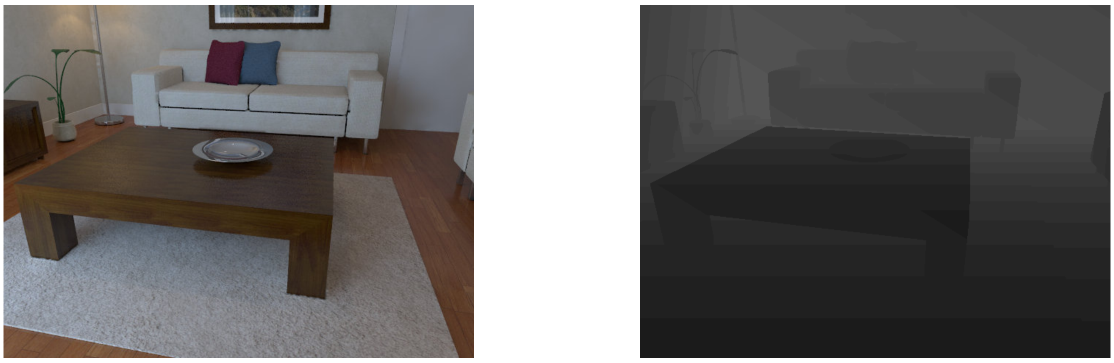


Figure 70: Test Images 34.

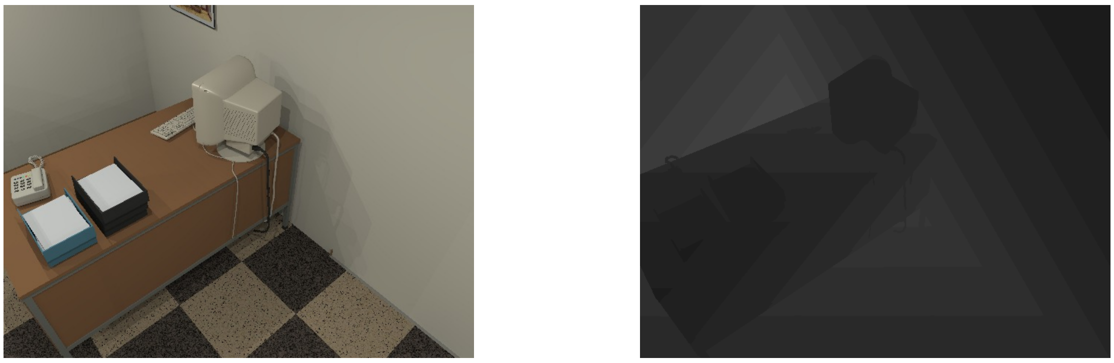


Figure 71: Test Images 35.

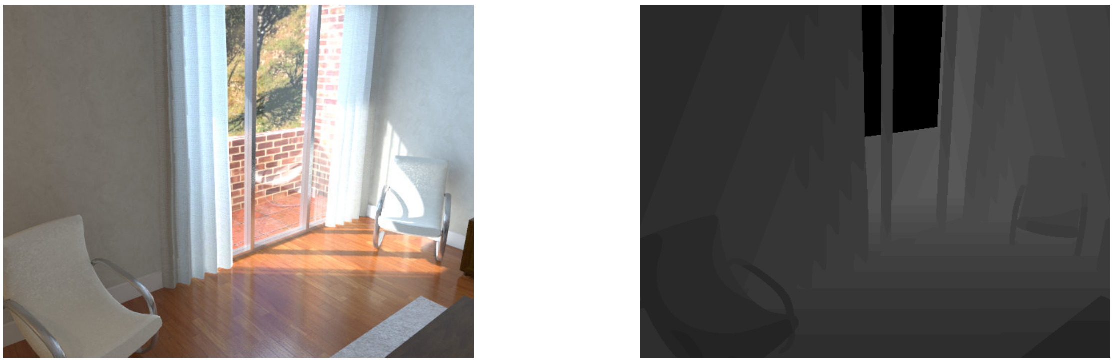


Figure 72: Test Images 36.

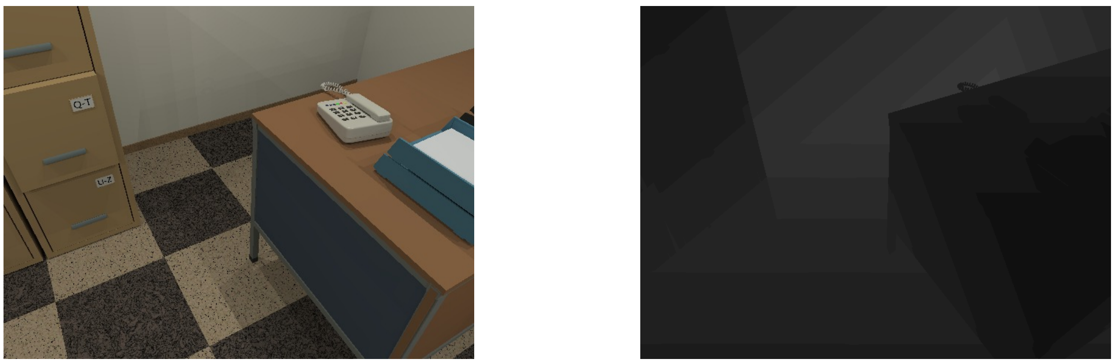


Figure 73: Test Images 37.

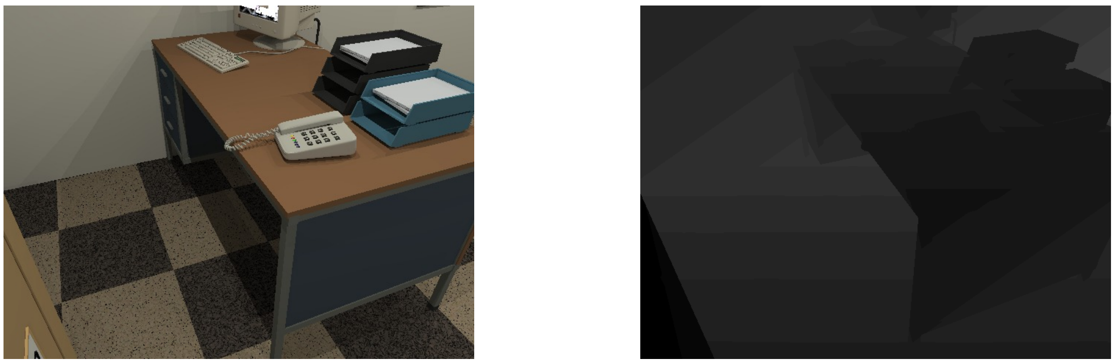


Figure 74: Test Images 38.

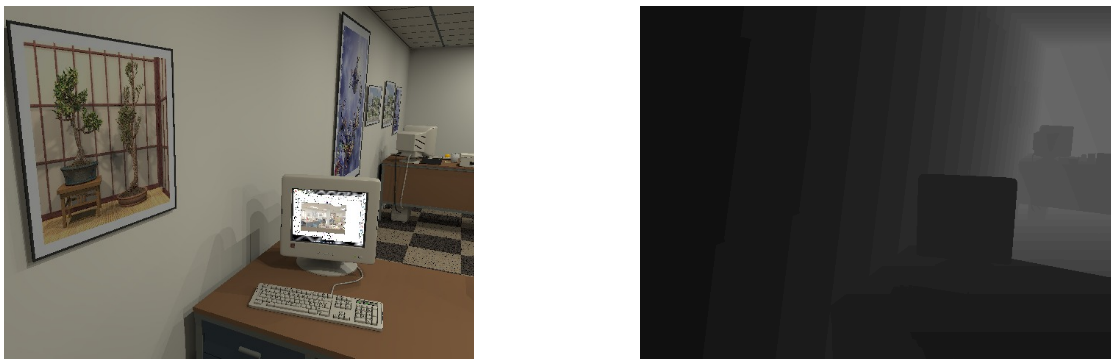


Figure 75: Test Images 39.

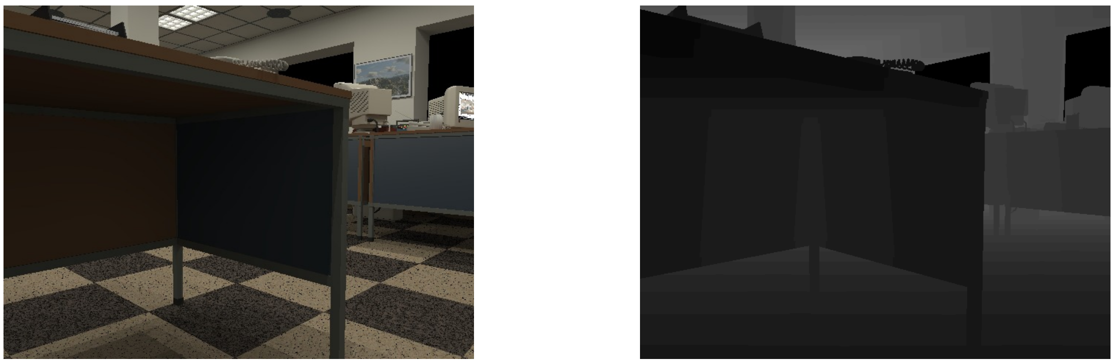


Figure 76: Test Images 40.

Printed by Tryckeriet i E-huset, Lund 2024

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2024-1006
http://www.eit.lth.se