Supplementary Material

May 15, 2023

1 Architecture of the proposed method

In this section, we will delve into the architecture of the proposed network in detail. This network is bifurcated into two main components.

The first part of the network ingests both the neighboring color images, denoted as $\{\mathbf{I}_i\}_{i=1}^{K}$, and raw depth maps, represented as $\{\mathbf{D}_i^{raw}\}_{i=1}^{K}$. The aim of this stage is twofold. Firstly, it completes the depth maps by filling in missing depth information. Secondly, it extracts critical visual and spatial features from the input images and depth maps. These features are essential as they contain information about the structure and appearance of the scene, which will be needed for the novel view synthesis.

Once the depth maps are completed and the visual and spatial features are extracted, these features are reprojected onto the target viewpoint according to the information provided by the completed depth maps. Finally, the reprojected visual features undergo a process of gradual fusion, guided by the reprojected spatial features. This fusion process integrates different visual features from various neighboring images, taking into account the spatial relationships among them. The purpose of this fusion is to create a seamless and coherent new view, minimizing artifacts and inconsistencies that may arise from the combination of different perspectives.

1.1 Depth completion and feature extraction modules

These modules take the color image and depth map of a nearby view as input. The subnetwork Φ is the feature encoder based on SwinTransformer, which is used to extract visual features. The subnetwork Θ, including several convolutional layers, is used for spatial feature extraction from the high-level visual features extracted by subnetwork Φ. The Υ module is responsible for integrating the spatial features extracted from both color images and depth maps. At each layer of this module, the corresponding features are concatenated and used to predict an attention vector, referred to as $\boldsymbol{\alpha}$. This vector guides the combination of features based on their relevance or importance. Finally, the Unet-decoder Γ is only used during the training process to generate the estimated depth map.

1.1.1 Visual feature extraction module

Figure 1 illustrates the architecture of the visual feature extraction module, comprising the Φ and Ω subnetworks based on SwinTransformer. The objective of the Φ subnetwork is to extract multi-scale visual features $\left\{\mathbf{F}_t^{\Phi}\right\}_{t=1}^{5}$ from the color images. These features serve two main purposes. Firstly, they provide visual information about the scene, ranging from coarse (low-level) to fine (high-level). SwinTransformer has demonstrated its effectiveness in various machine vision tasks, such as monocular depth estimation, indicating that the high-level features it extracts can convey information about the scene's structure and object characteristics. In this study, we leverage these features to estimate additional spatial features, which are then combined with spatial features from the depth maps to generate complete depth maps.

The second goal of the multi-scale visual features is to capture the textures of local areas for the subsequent view synthesis module. To achieve this, we progressively merge the features into a single feature map $\mathbf{F}_i^c$ (where i is the index of the neighboring view) with the same resolution as the input color image using the Ω subnetwork. This enables the feature vector for each pixel to describe the visual textures over a large local area. Having a feature vector that encompasses a larger local area offers two advantages. Firstly, it assists

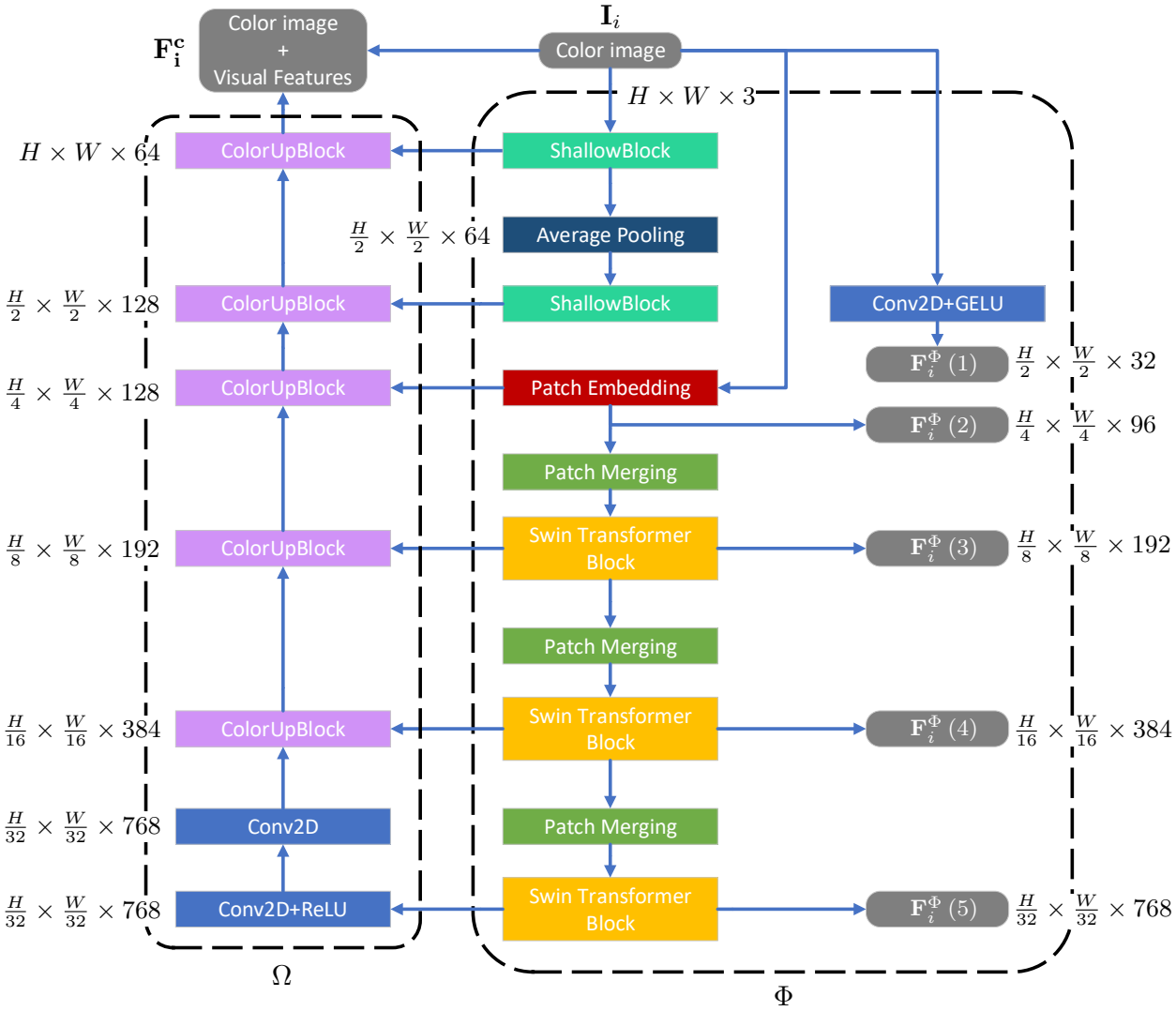


Figure 1: The image depicts the visual feature extraction subnetwork.

the synthesis network in filling in the gaps created during reprojection. Secondly, it provides supplementary information for the network to address conflicts by maximizing texture consistency.

The subnetwork Φ is comprised of three branches. The first branch is the SwinTransformer feature extraction component, which includes three key parts: PatchEmbedding, PatchMerging, and the Swin Transformer Block. The PatchEmbedding process is the initial step in this architecture, which divides the input image into numerous small patches and then embeds them into a higher-dimensional space. This step is crucial as it transforms the original image data into a format that can be processed by the subsequent transformer layers. However, it should be noted that the output feature maps of the PatchEmbedding are four times smaller in resolution compared to the original image. To address this, two additional branches are constructed to extract feature maps of a higher resolution. The architecture of ShallowBlock and ColorUpBlock are illustrated in Figure 2.

1.1.2 Depth completion module

The depth completion module in our work consists of three subnetworks: Θ, Υ, and Ψ. The Θ subnetwork takes in the multi-scale visual features $\left\{\mathbf{F}_t^\Phi\right\}_{t=1}^5$ and generates multi-scale spatial features, as mentioned

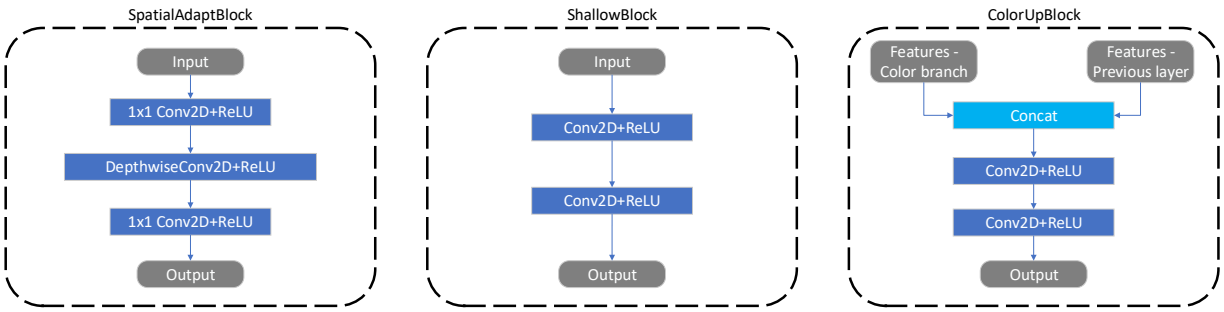


Figure 2: The image depicts different build blocks of the proposed networks.

earlier. The Θ subnetwork is based on the SpatialAdaptBlock (see Figure 2).

Additionally, we employ the Ψ subnetwork, which follows a Unet-based architecture, to extract spatial features from the depth map. These extracted features are then combined with the spatial features from the color image. The gradual fusion of these spatial features from both sources is performed by the Υ subnetwork. The construction of the Υ subnetwork is based on the FusionBlock (see Figure 4).

Each fusion block receives the spatial feature from the color branch $\mathbf{F}_i^{\Theta}(t)$, the output of the previous layer $\mathbf{F}_i^{\Upsilon}(t-1)$ in the Υ subnetwork, and the spatial feature from the depth branch $\mathbf{F}_i^{\Psi}(t)$. The main idea of the fusion block is to predict the confidence map ($\boldsymbol{\alpha}$) for the spatial features extracted from the depth map. Subsequently, the spatial features are merged based on this confidence map and upscaled accordingly. Finally, the completed depth map is computed at the end of the Υ subnetwork using a single convolutional layer followed by a sigmoid activation. Additionally, the Υ subnetwork outputs the spatial features extracted from the last FusionBlock. These spatial features provide the geometrical information necessary to guide the fusion of the reprojected visual features in subsequent steps.

1.2 Novel view synthesis module

After obtaining the completed map, we can estimate the mapping between the source viewpoints and the target viewpoint (the extrinsic and intrinsic parameters of the cameras are already known). In this work, we reproject both the visual features $\{\mathbf{F}_i^c\}_{i=1}^K$ and spatial features $\left\{\mathbf{F}_i^d\right\}_{i=1}^K$ to the target viewpoint, resulting in the reprojected visual features $\{\mathbf{F}_i^{pc}\}_{i=1}^K$ and spatial features $\left\{\mathbf{F}_i^{pd}\right\}_{i=1}^K$, respectively. Then, the reprojected features are fed into the ConvGRUUNet-based network to fuse the features and generate the novel view images.

The architectures of the encoder and decoder of the ConvGRUUNet-based network are depicted in Figure 7 and Figure 8, respectively. For each input image (from a neighboring view), the network output a confidence map $\mathbf{M}_i$ and an output image $\mathbf{I}_i$. The final output of the network is fused as shown in Equation 1, where $\circ$ denotes the element-wise multiplication. Since different camera viewpoints may observe the reflected light differently, we apply the weighted average instead to minimize texture errors. We show the example confidence maps in Figure 5 and the synthesized image at each step in Figure 6.

$$\hat{\mathbf{I}} = \frac{\sum_{i=0}^{K} \mathbf{M}_i \circ \mathbf{I}_i}{\sum_{i=0}^{K} \mathbf{M}_i} \tag{1}$$

2 Optimization

The datasets we utilized to assess the effectiveness of the novel view synthesis techniques do not include perfect ground-truth depth maps, but only raw depth maps. Consequently, during the training phase, we randomly introduce gaps into the raw depth maps and then feed the resulting augmented depth maps to the network as input. The raw depth maps are utilized as the training ground-truth, which compels the

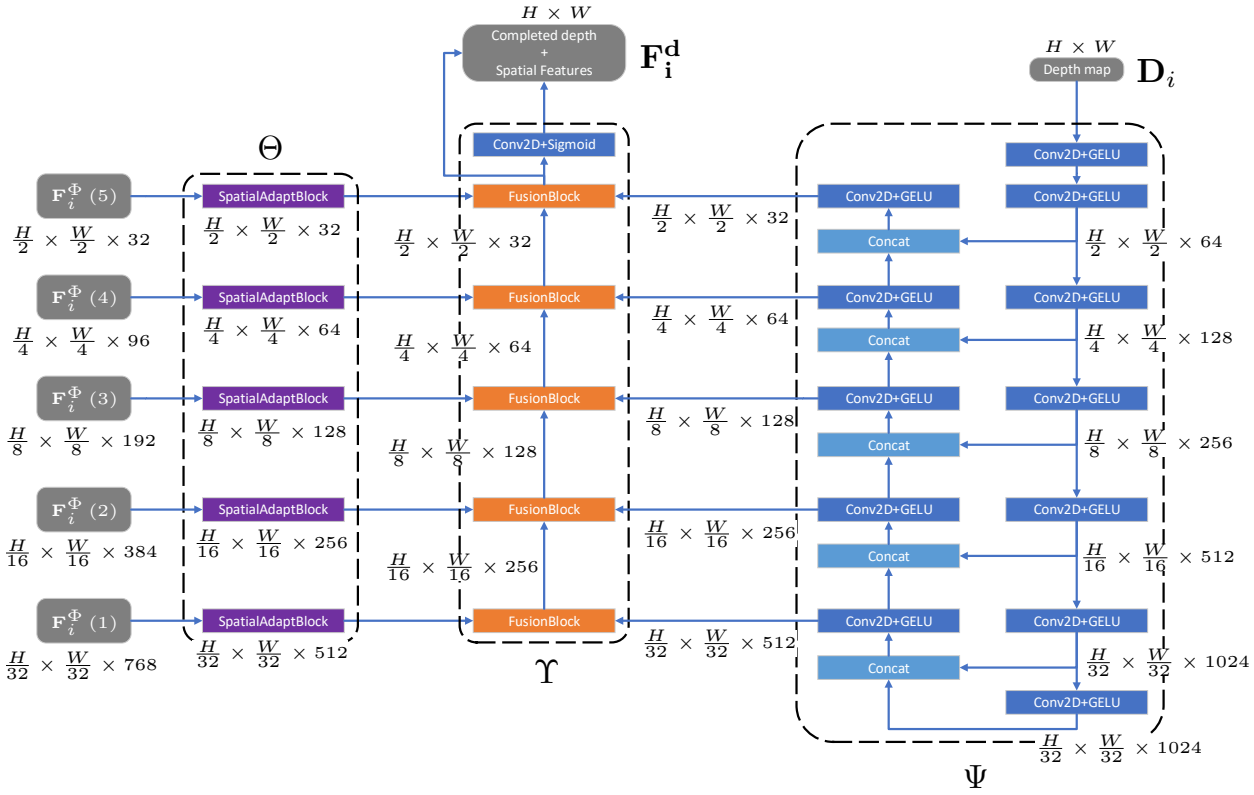


Figure 3: The image depicts the depth completion and feature extraction modules. Note that, the Unet-encoder of Φ has been replaced with SwinTransformer.

network to fill in the artificial gaps that were introduced into the input images. Given that the DTU dataset comprises a low number of training samples, hence, we randomly alter the brightness, contrast, saturation, and hue of both the input and target color images in order to generate more training samples.

The network's modules are trained simultaneously using the losses specified in Equation 2. The output of the network is denoted as $\hat{\mathbf{I}}$, while the ground-truth image of the target viewpoint is represented as $\mathbf{I}$. Depth maps from neighboring viewpoints are given by $\mathbf{D}^{raw}$, completed depth maps by $\hat{\mathbf{D}}^{com}$, and estimated depth maps by $\hat{\mathbf{D}}^{est}$. The loss $\mathcal{L}_{depth}$ measures the accuracy of the estimated and completed depth maps against the input depth maps, while the loss $\mathcal{L}_c$ assesses the error between the synthesized image and the target image.

$$\mathcal{L} = \mathcal{L}_{depth}(\hat{\mathbf{D}}_k^{est}, \hat{\mathbf{D}}_k^{com}, \mathbf{D}_k^{raw}) + \lambda_{vis}\mathcal{L}_{vis}\left(\hat{\mathbf{I}}, \mathbf{I}\right) \tag{2}$$

We show the details to compute $\mathcal{L}_{depth}$ in Equation 3 and Equation 4, which is based on the depth loss in [1]. The $\mathcal{L}_d$ loss function comprises two parts within the square root. The first part, which is the mean square error, optimizes the accuracy of the enhanced depth maps. The second part, known as the consistent loss, preserves the shape of objects. Additionally, we calculate the loss using normalized depth values. Since depth map errors can have different impacts at different distances, we address this issue by calculating the loss function based on the natural logarithm.

$$\mathcal{L}_{depth}(\hat{\mathbf{D}}_k^{est}, \hat{\mathbf{D}}_k^{com}, \mathbf{D}_k^{raw}) = \frac{\lambda_{est}\Sigma_{k=1}^{K}\mathcal{L}_d\left(\hat{\mathbf{D}}_k^{est}, \mathbf{D}_k^{raw}\right)}{K} + \frac{\lambda_{com}\Sigma_{k=1}^{K}\mathcal{L}_d\left(\hat{\mathbf{D}}_k^{com}, \mathbf{D}_k^{raw}\right)}{K} \tag{3}$$

$$\mathcal{L}_d(\hat{\mathbf{D}}_k, \mathbf{D}_k^{raw}) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} h^2\left(\hat{\mathbf{D}}_k, \mathbf{D}_k^{raw}, n\right) - \frac{\lambda_\sigma}{N^2}\left[\sum_{n=1}^{N} h\left(\hat{\mathbf{D}}_k, \mathbf{D}_k^{raw}, n\right)\right]^2} \tag{4}$$

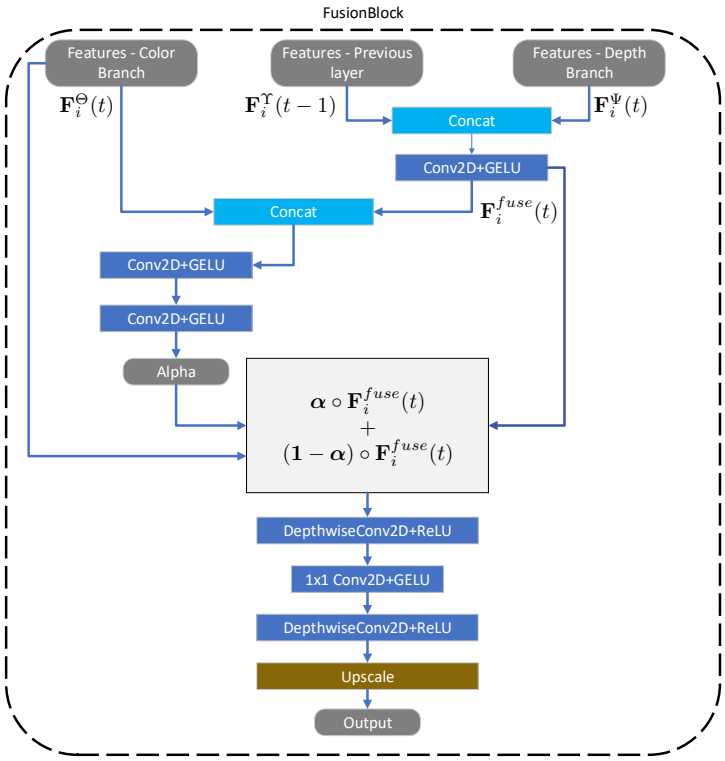


Figure 4: The image depicts architecture of the FusionBlock.

(a) View 1 (b) View 2 (c) View 3

Figure 5: The images illustrate the estimated confidence maps generated by the novel view synthesis module.

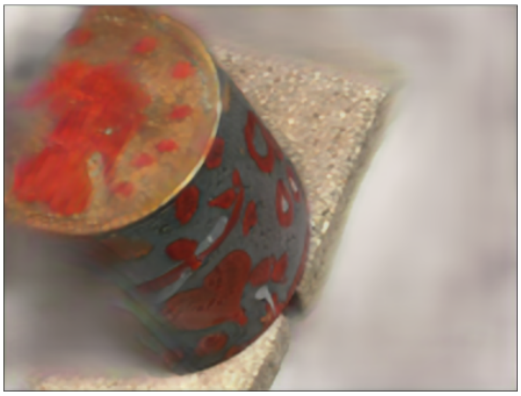
where $h\left(\hat{\mathbf{D}}_k, \mathbf{D}_k^{raw}, n\right) = \ln\left(\hat{\mathbf{D}}_k\left(n\right)\right) - \ln\left(\mathbf{D}_k^{raw}\left(n\right)\right)$, N represents the total number of pixels in the images, and K represents the total number of neighboring cameras.

Similar to [2], we compute the loss $\mathcal{L}_{vis}$ between the synthesized image and the target image based on L2 and perceptual losses $\mathcal{L}_c$ [3].

$$\mathcal{L}_{vis}\left(\hat{\mathbf{I}}, \mathbf{I}\right) = \lambda_{l_2}\mathcal{L}_{l_2}\left(\hat{\mathbf{I}}, \mathbf{I}\right) + \lambda_c\mathcal{L}_c\left(\hat{\mathbf{I}}, \mathbf{I}\right) \tag{5}$$

3 Experimental setup

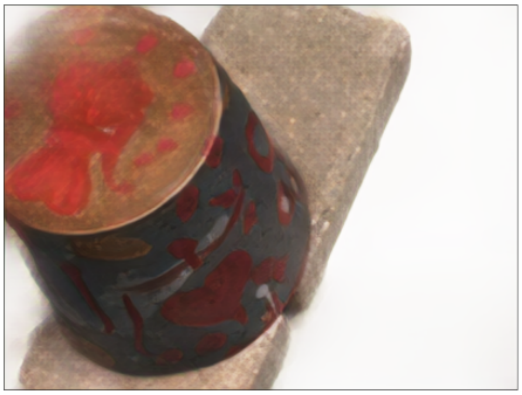
In this work, all experiments were conducted on an Nvidia GeForce GTX 1080 Ti with 11 GB of GDDR5X memory to ensure fair comparisons. The model was implemented using PyTorch 1.6.0 with Python 3.7.13 and optimized using the Adam optimizer [4]. Standard initial learning rate (10^{-4}), exponential decay rates



(a) View 1

(b) View 2

(c) View 3



(d) Merged

Figure 6: The images show the illustration of the output by the novel view synthesis module for each step (for each neighboring viewpoint). As you can see, each image has slightly different textures on the object. So, we can achieve better output image by using the weighted average fusion.

($\beta_1 = 0.9$, $\beta_2 = 0.999$), and weight decay rate ($\lambda = 0.01$) were employed, as they have been proven effective in various deep learning applications.

References

[1] Ziheng et al. Li. Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation. *Technical Report arXiv:2203.14211*, 2022.

[2] Alan et al. Cao. FWD: Real-time novel view synthesis with forward warping and depth. In *CVPR*, pages 15692–15703, New Orleans, USA, 2022.

[3] J. et al. Johnson. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, Amsterdam, Netherlands, 2016. Springer International Publishing.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Technical Report arXiv:1412.6980*, 2015.

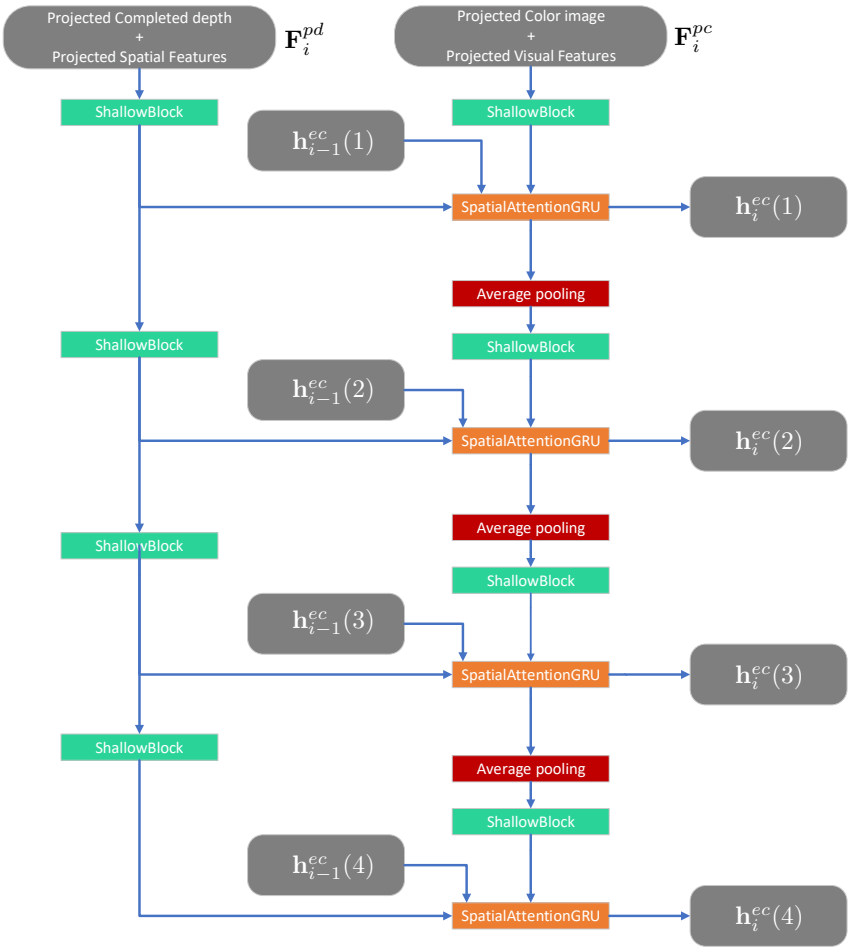


Figure 7: The image depicts the encoder of the novel view synthesis module of the proposed method.

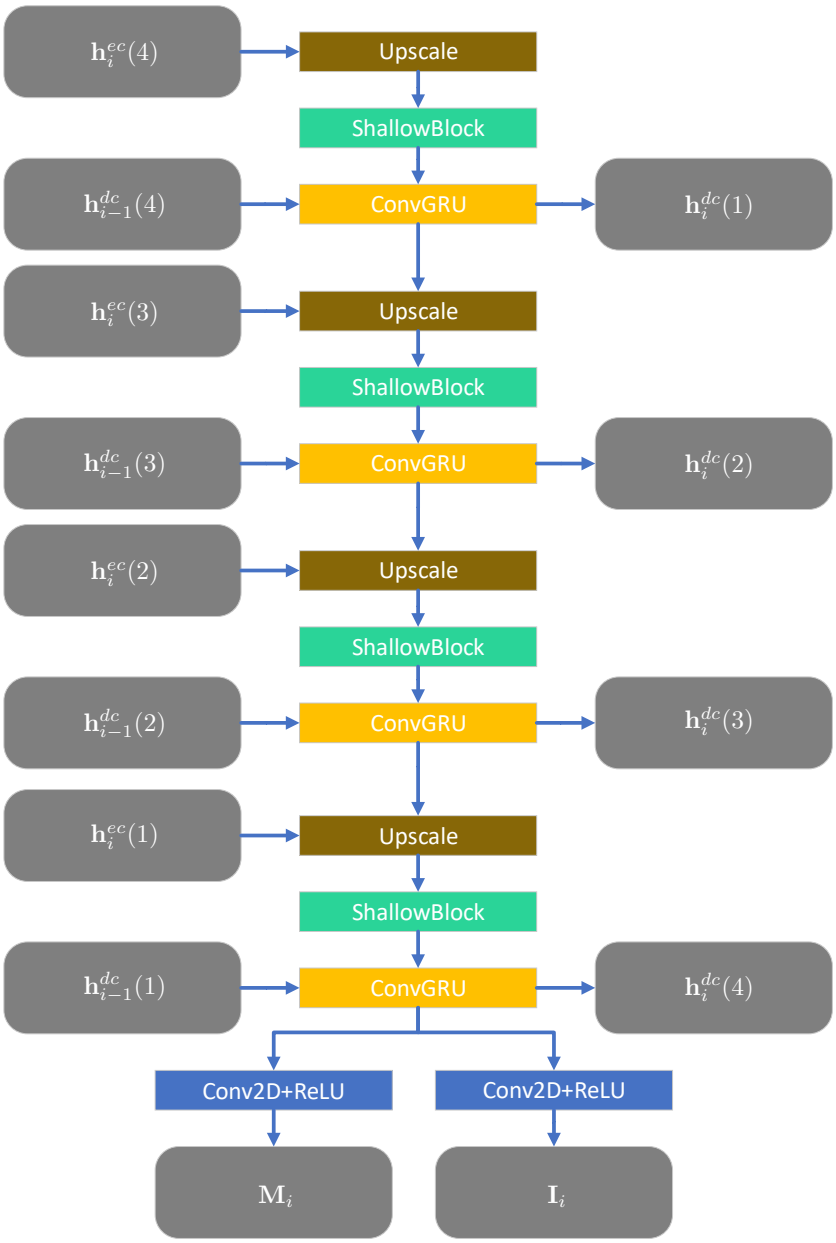


Figure 8: The image depicts the decoder of the novel view synthesis module of the proposed method.