Pyramidal Layered Scene Inference with Image Outpainting for Monocular View Synthesis*

Marcos R. Souza[1][0000-0003-4342-5220], Jhonatas S. Conceição[1][0000-0001-9414-3341], Jose L. Flores-Campana[1][0000-0001-5461-5869], Luis G. L. Decker[1][0000-0002-6959-3890], Diogo C. Luvizon[2][0000-0002-5055-500X], Gustavo Sutter P. Carvalho[2][0000-0002-4275-5255], Helena A. Maia[1][0000-0002-8253-9004], and Helio Pedrini[1][0000-0003-0125-630X]

[1] Institute of Computing, University of Campinas, Campinas, SP, Brazil, 13083-852
[2] AI R&D Lab, Samsung R&D Institute Brazil, Campinas, SP, Brazil, 13097-160

Abstract. Generating novel views from a single input is a challenging task that requires the prediction of occluded and non-visible content. Nevertheless, it is an interesting and active area of research due to its several applications such as entertainment. In this work, we propose an end-to-end architecture for monocular view synthesis based on the layered scene inference (LSI) method. The LSI uses layered depth images that can represent complex scenes with a reduced number of layers. To improve the LSI predictions, we develop two new strategies: (i) a pyramidal architecture that learns LDI predictions for different resolutions of the input and (ii) an image outpainting for filling the missing information at the LDI borders. We evaluate our method on the KITTI dataset, and show that the proposed versions outperform the baseline.

Keywords: Monocular View Synthesis · Layered Depth Image · Pyramidal Network · Image Outpainting

1 Introduction

The monocular view synthesis aims to produce images from different viewpoints of the scene using a single image as input. Either implicitly or explicitly, this task involves interpreting complex structures in the scene through texture and depth, and filling the content that is not visible in the original viewpoint.

This problem can benefit several other tasks, such as augmented reality systems. A very interesting application is the generation of parallax motion effect [8, 12, 21], in which a sequence of new views can be created from a single source view. When we see the entire sequence, the objects close to the observer must have a higher perceived velocity than the farther deep objects.

* This work was funded by Samsung Eletrônica da Amazônia Ltda., through the project "Parallax Effect", within the scope of the Informatics Law No. 8248/91.

Recent work has used deep architectures to predict different representations such as 3D meshes [9] or point clouds [20]. Two other approaches have been widely explored to represent 3D scenes in complex images: multiplane image (MPI) [4, 10, 15, 17, 22] and layered depth images (LDI) [2, 3, 14, 18]. According to Shih et al. [14], MPI may produce artifacts on sloped surfaces and contain redundant information across its layers. Moreover, the high number of layers typically used in MPIs leads to a high computational cost. For this reason, here we explore LDI for a compact representation of the scene.

LDI represents a 3D scene by a layered representation. It was originally proposed for image-based rendering [13] and defined as "a view of the scene from a single input camera view, but with multiple pixels along each line of sight". As observed by the authors, the size of the LDI grows linearly with the observed depth complexity in the scene. Each layer of the LDI consists of a 4-channel image, in which the first three channels are the RGB information, and the last one is the corresponding disparity map. The first layer represents the visible content from the original viewpoint and, so its RGB channels correspond to the input image. From the second layer onwards, the LDI stores the information that was occluded in the first layer.

Some recent work aims to calculate new views from calibrated stereo images [4, 15, 22]. An even challenging approach is the prediction of new views from a single view [3, 10, 12, 17, 18]. Most of these works use deep networks to solve intermediate tasks, but do not build a deep end-to-end model for the final task (new view generation). An exception is the layered scene inference (LSI) method [18], in which LDIs are calculated by an end-to-end network from either monocular or stereo-based datasets. This type of strategy has the advantage that intermediate tasks can benefit from the final supervision, which allows the network to find the representation that best minimizes the final loss function.

In this work, we propose a set of incremental strategies based on LSI [18] to improve the LDI prediction through deep end-to-end networks. More specifically, we focus on (a) reducing the distortions present at the rendition boundaries and (b) improving the overall rendering quality (similarity with the ground-truth). To address these issues, we propose and analyze the use of (i) an outpainting method to extrapolate the boundaries of the LDI and (ii) a pyramidal architecture, which learns how to compute LDIs at multiple resolutions.

2 Proposed Method

In this section, we present our method and compare it with the baseline LSI [18]. Both methods are illustrated in Figure 1. The baseline uses an encoder-decoder architecture, the DispNet [11], to predict LDI from an RGB source image. Then a differentiable rendering is applied to compute the target image.

LSI [18] was trained with a set of N inputs $(I_s^n, I_t^n, K_s^n, K_t^n, R^n, t^n)_{n=1}^N$. Images I_s^n and I_t^n are respectively the source image and an arbitrarily sampled target, both from the same scene. Matrices K_s^n and K_t^n define the intrinsic pa-

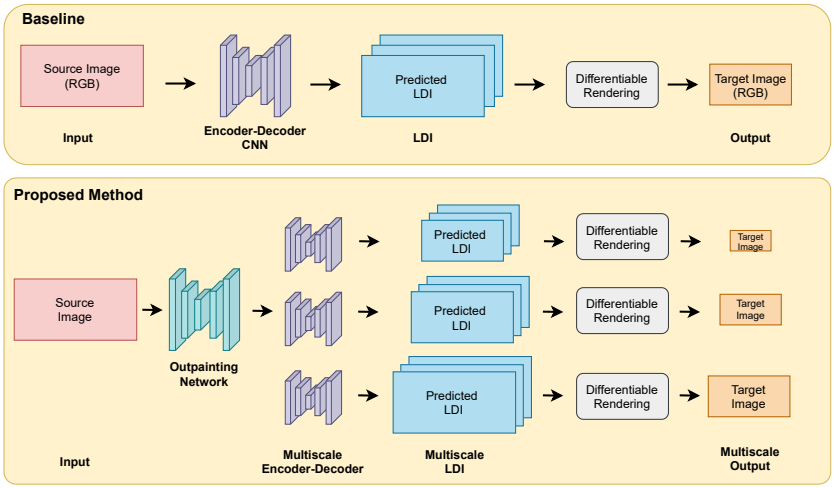


Fig. 1: Overview of the baseline and our proposed method.

rameters of the cameras that captured the two images. Finally, the matrices R^n and t^n describe the relative rotation and translation between both cameras.

An LDI representation with L layers is created from each source image I_s^n. It can be described as $(I^l, D^l)_{l=1}^{L}$, where I^l e D^l are images that represent respectively the texture and disparity of each pixel from I_s^n for the l-th layer. The disparity maps must satisfy the condition $D^l(p) \leq D^{l+1}(p)$, that is, the disparity of the same pixel cannot be greater in a deeper layer. The reason for this condition is that, if the texture of an object took many layers to become visible, it is farther away from the camera or covered by many objects. Therefore, it should have a smaller disparity (greater depth).

From the input camera parameters and a predicted LDI, the LSI can project the source on the target and compute the rendered texture using a differentiable soft z-buffering. Then, the network is supervised by a set of five loss functions: (i) the view synthesis loss $\mathcal{L}_{\text{vs}}$ that compares the rendered and ground-truth targets; (ii) the 'min-view synthesis' loss $\mathcal{L}_{\text{m-vs}}$ used to improve the background layers contribution; (iii) the source consistency loss $\mathcal{L}_{\text{sc}}$ given by the L_2 norm weighted by the disparity map; (iv) the depth monotonicity loss $\mathcal{L}_{\text{inc}}$ that ensures a non-increasing disparity; and (v) the smoothness loss $\mathcal{L}_{\text{sm}}$ given by the L_1 norm of the second-order spatial derivatives of the predicted disparity maps. The loss functions $\mathcal{L}_{\text{vs}}$ and $\mathcal{L}_{\text{m-vs}}$ are based on a mask M defined to ignore the image boundaries, since these pixels may be out of the field of view in the source image and, therefore, the network does not have the required information for a reliable prediction of them. This aspect will be further discussed in Subsection 2.1.

From these five loss functions, the final objective function $\mathcal{L}_{\text{final}}$ is defined as

$$\mathcal{L}_{\text{final}} = \lambda_{\text{vs}}\mathcal{L}_{\text{vs}} + \lambda_{\text{m-vs}}\mathcal{L}_{\text{m-vs}} + \lambda_{\text{sc}}\mathcal{L}_{\text{sc}} + \lambda_{\text{inc}}\mathcal{L}_{\text{inc}} + \lambda_{\text{sm}}\mathcal{L}_{\text{sm}}, \tag{1}$$

where λ_x defines the weight of each loss function $\mathcal{L}_{\text{x}}$.

Our method introduces two new strategies in the baseline (Figure 1): (i) the outpainting network, detailed in Subsection 2.1, that extrapolates the input image borders and (ii) the pyramidal approach (Subsection 2.2), in which we use the prediction from the immediately lower scale to support the current prediction. The pipeline in each level is similar to the LSI, except for the outpainting step. As the original, our network is trained in an end-to-end fashion.

2.1 Outpainting

As previously mentioned, the baseline uses loss functions that ignore the target image boundaries. This may lead to a poor prediction in that region. To circumvent this, we extrapolate the source image so that the LDI has extra information at the image borders. These extra pixels will be used in rendering the new images, allowing complete supervision of the target image. Other works in the literature have already used outpainting as a strategy to extrapolate the field of view of the representation to be rendered [14].

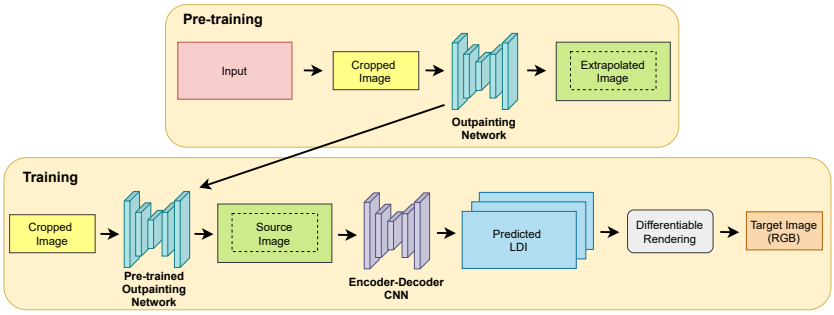


Fig. 2: Overview of the outpainting strategy.

For the outpainting, we use an encoder-decoder network recently proposed [19], which is trained with pixel-wise and adversarial losses. Initially, the outpainting network is pre-trained separately, instead of training the model from scratch along with the LDI predictor. After the pre-training, we use the weights of the outpainting network in the equivalent layers of ours, which is then trained keeping the outpainting layers frozen. This process is illustrated in Figure 2.

The leftmost image in the pre-training figure is the original one. During the pre-training, we crop their boundaries removing some pixels and the network extrapolates the cropped image, producing a new image with the same resolution as the original one. In the training, the input images are also cropped and extrapolated by the pre-trained model. This new boundary added to the source gives extra information to render the target. Thus, although the target still has half the input size (i.e., cropped image size), it is built from a broader field of view thanks to the extrapolation. For this reason, we modify the loss functions

$\mathcal{L}_{\text{vs}}$ and $\mathcal{L}_{\text{m-vs}}$ to consider the entire images in the supervision, which are now described as

$$\mathcal{L}_{\text{vs}} = \|I_t - \bar{I}_t\|_1, \qquad \mathcal{L}_{\text{m-vs}} = \sum_{p_t} \min_l \|I_t(p_t) - \bar{I}_t^l(p_t)\|_1, \tag{2}$$

where I_t and $\bar{I}_t$ are the original and predicted target image, and $\bar{I}_t^l$ is the predicted image using only the l-th layer. To train the outpainting network, we use the loss $\mathcal{L}_{\text{sc}}$ defined in Equation 3, where I_s is the ground-truth image for outpainting, $D^o()$ is the discriminator loss for the extrapolated image I_s^o, and the λ^o is a constant weight. In the LSI network, we use the entire images as inputs without the cropping step.

$$\mathcal{L}_{\text{sc}}^o = \sum_{p_s} \|I_s(p_s) - I_s^o(p_s)\|_2 + \lambda^o D^o(I_s, I_s^o). \tag{3}$$

2.2 Pyramidal Network Architecture

Our second proposal is the use of a pyramidal architecture. Pyramidal architectures [6] have been successfully used in diverse types of related problems, such as depth estimation [1] and optical flow estimation [16]. This architecture predicts a low resolution LDI and uses this prediction to compute the LDI at the next highest resolution. Figure 3 presents a diagram of this strategy. The architecture adopted for this task was inspired by the pyramidal PWC-Net [16] proposed for optical flow prediction.

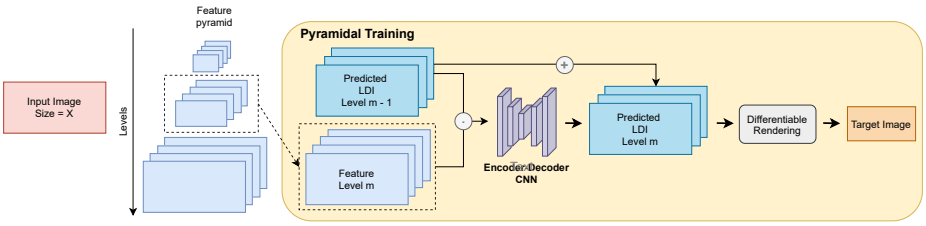


Fig. 3: Representation of the pyramidal architecture.

As seen in the figure, we built a pyramid of features extracted from the input image. It uses a convolutional network with learnable weights, that are learned in the end-to-end training. According to Sun et al. [16], the main advantage of the learnable feature pyramid over an image pyramid is that the former is robust to shadows and lighting changes.

The network predicts an LDI representation for each level using the respective feature as input. To feed the encoder-decoder CNN, this feature is concatenated with the last LDI. The last LDI is bilinearly interpolated to match the

feature size. A residual connection combines every representation into a single one. Therefore, the network learns the complement of the LDI at each level. The only exception is the first level of the pyramid, in which we do not use either the concatenation or the residual connection, since no LDI representation is yet available. The prediction of the LDI in each level uses a coarse-to-fine strategy. After computing the coarse LDI (as the complement of the last LDI), it feeds another CNN in order to predict the fine LDI.

The architecture illustrated in Figure 3 is an end-to-end trained network, including the layers responsible for the construction of the feature pyramid and the encoder-decoder CNN. The new final loss function comprising all pyramid levels is described as

$$\mathcal{L}_{\text{final}} = \frac{\sum_m^M \lambda_{\text{vs}}\mathcal{L}_{\text{vs}}^m + \lambda_{\text{m-vs}}\mathcal{L}_{\text{m-vs}}^m + \lambda_{\text{sc}}\mathcal{L}_{\text{sc}}^m + \lambda_{\text{inc}}\mathcal{L}_{\text{inc}}^m + \lambda_{\text{sm}}\mathcal{L}_{\text{sm}}^m}{M} \tag{4}$$

where M is the number of pyramid levels and $\mathcal{L}_x^m$ is the loss components calculated specifically for the m-th pyramid level.

3 Results

In this section, we show and discuss our results using the raw KITTI [5], which contains videos captured by a recording platform equipped with stereo cameras and other sensors that provide diverse information such as geographic coordinates. The reader may refer to the dataset paper for further details about the cameras and recording setup. As in the LSI [18], we randomly pick the right or left image as the source and the other one as the target. We use 33 sequences and multiple image pairs from each sequence, resulting in 22600 samples for training and 888 for validation.

Our method was implemented using PyTorch for deep learning, based on the TensorFlow LSI implementation provided by Tulsiani et al. [18]. The results of the reimplementation were very similar to the ones reported by the authors. The lambda weights of Equations 1 and 4 were 1, 1, 25, 25 and 0.65, respectively. The higher weights were assigned to $\mathcal{L}_{\text{sc}}$ and $\mathcal{L}_{\text{inc}}$ to correct the discrepancy among the losses. For the outpainting-based versions, the λ_{sm} was set to 1. We performed a standard data augmentation, with brightness, contrast, and saturation adjustment. The images were normalized in the range of $[0, 1]$ and resized to 512×256 pixels. The entire network was trained for 40 epochs. The initial learning rate was set to 1e-3 for LSI and 2e-5 for the pyramidal network. The learning rate scheduler multiplies it by 0.1 at every 20 epochs.

The PyTorch implementation of the outpainting network was provided by Hoorick [7]. The outpainting pre-training was done using the same training samples as the entire network. The training loss was given in Equation 3. It was trained for 130 epochs. The value of λ^o was set to 0.001, 0.005, 0.015 and 0.040, in the epochs 1, 10, 30 and 60, respectively.

Table 1 presents the results of the different versions of our method compared to the baseline using the pixel-wise L_1 error and the Structural Similarity

(SSIM) metric. In addition to the traditional L_1 error, we also show I-L_1 and O-L_1, which consider only inner (I-L_1) or outer (O-L_1) pixels to compute the metric. For them, we considered a border of 10% in each side, based on the mask M used in the original LSI. Similarly, we also compare the methods using inner- and outer-SSIM. In the table, LSI, LSI+IO and PyLSI refer respectively to the PyTorch baseline, the LSI with the image outpainting network and the pyramidal LSI. Besides the sequential combination of outpainting and pyramidal strategy presented in Figure 1 referred to as SeqLSI, we also tested a parallel version (ParLSI) with an offline late fusion. This fusion blends the rendering of two methods (Figures 2 and 3). The inner and outer pixels of the final rendering are directly computed from the rendering of the pyramidal and the outpainting networks, respectively. To avoid discontinuities, we used a blending mask smoothed by a Gaussian filter.

Table 1: $L_1 \downarrow$ error and SSIM $\uparrow$ on the validation set of the KITTI dataset.

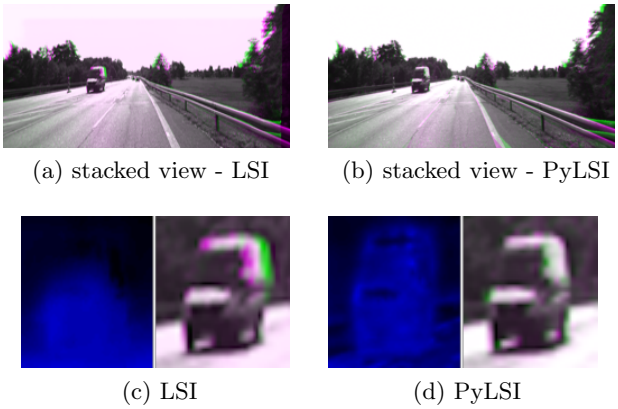
1-layer LDI				**2-layer LDI**			
Version	**L_1**	**I-L_1**	**O-L_1**	**Version**	**L_1**	**I-L_1**	**O-L_1**
LSI [18]	0.0613	0.0539	0.0746	LSI [18]	0.0609	0.0552	0.0710
LSI+IO	0.0533	0.0512	**0.0571**	LSI+IO	0.0670	0.0655	0.0696
PyLSI	0.0482	**0.0396**	0.0643	PyLSI	**0.0597**	**0.0549**	0.0686
SeqLSI	0.0547	0.0547	0.0588	SeqLSI	0.0599	0.0582	**0.0629**
ParLSI	**0.0458**	0.0421	0.0604	ParLSI	0.0606	0.0596	0.0646
Version	**SSIM**	**I-SSIM**	**O-SSIM**	**Version**	**SSIM**	**I-SSIM**	**O-SSIM**
LSI [18]	0.7289	0.7584	0.5507	LSI [18]	0.7223	0.7499	0.5489
LSI+IO	0.7208	0.7388	0.6217	LSI+IO	0.7034	0.7208	0.6168
PyLSI	0.7567	0.7724	0.5946	PyLSI	0.7424	0.7575	0.5810
SeqLSI	0.7037	0.7195	0.6145	SeqLSI	0.7528	0.7711	**0.6556**
ParLSI	**0.7798**	**0.8047**	**0.6309**	ParLSI	**0.7687**	**0.7931**	0.6350

We can see from Table 1 that the PyLSI outperformed the baseline LSI in all cases. This is more evident in the 1-layer LDI, where the I-L_1, for instance, reaches 0.0396, a 26% drop compared to the LSI. Although the 2-layer PyLSI has also improved the LSI, it was not so intense as in the 1-layer, leading to a decrease of only 0.5% in I-L_1. As expected, the LSI+IO achieved good results in outer pixels, where the LSI has its worst results. In the 1-layer LDI, the O-L_1 was closer to the other metrics and presented a drop of about 23% compared to the LSI. The O-SSIM was increased by around 13%.

The numerical results of the 1-layer SeqLSI were not as good as those achieved by the isolated versions in inner pixels (PyLSI) and outer pixels (LSI+IO), although it did outperform the baseline in most cases. However, the 2-layer version obtained better SSIM values in all regions of the image. On the other hand, 1-

layer ParLSI achieved the best results for L_1 and all SSIM values, whereas the 2-layer version obtained the best results on SSIM.

Figure 4 presents a comparison between the results of LSI and PyLSI. Subfigures 4a and 4b show the new views of LSI and PyLSI, respectively, stacked with the target image on the RGB channels. On these stacked images, we expect values close to gray when the prediction is good and close to green or purple when it is not. Finally, Subfigures 4c and 4d present a zoomed region of the disparity and stacked images of the LSI and the PyLSI, respectively.



(a) stacked view - LSI (b) stacked view - PyLSI

(c) LSI (d) PyLSI

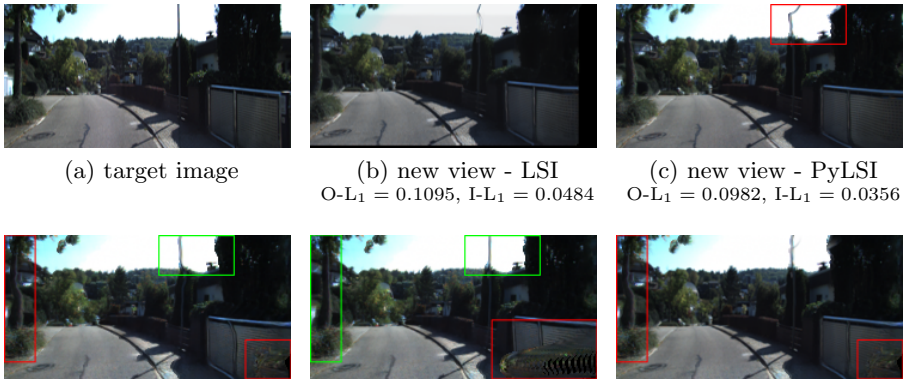
Fig. 4: Comparison between LSI ($L_1 = 0.0683$) and PyLSI results ($L_1 = 0.0328$).

On an overview of the stacked images, we can see that the PyLSI version has values closer to gray, even in more homogeneous regions, such as the sky. This shows that the intensities and colors of the PyLSI prediction are closer to the ground-truth than the LSI prediction. In addition, as we can see in the zoomed images, the PyLSI disparity in the car is better defined, which leads to better alignment at the prediction.

Figure 5 presents another example for all versions. Comparing the views rendered by LSI and PyLSI, we can see an improvement in the image boundaries. However, as there is no extrapolation and supervision on these regions, they are not reliable and lead to distortions in the objects (highlighted). The outpainting improves these distortions but generates some artifacts in the three versions that include it (LSI+IO, SeqLSI and ParLSI). These artifacts could be removed using a more recent and advanced outpainting network. Despite the artifact, the best visual result is achieved by SeqLSI.

4 Conclusions

In this work, we presented a new method for monocular view synthesis based on LSI. Our method is composed of an initial outpainting step and a pyramidal



(a) target image

(b) new view - LSI
O-L_1 = 0.1095, I-L_1 = 0.0484

(c) new view - PyLSI
O-L_1 = 0.0982, I-L_1 = 0.0356

(d) new view - LSI+IO
O-L_1 = 0.0933, I-L_1 = 0.0331

(e) new view - SeqLSI
O-L_1 = 0.0915, I-L_1 = 0.0361

(f) new view - ParLSI
O-L_1 = 0.0913, I-L_1 = 0.0288

Fig. 5: Example of results for all versions.

strategy. The main goal of the outpainting network was to increase the original field of view and improve the prediction in the boundaries of the new view. The pyramidal strategy was employed to improve the overall quality by generating a new view for different resolutions of the input. We tested the two strategies separately (LSI+IO and PyLSI), and combined them sequentially (SeqLSI) and in parallel (ParLSI). We compared them with the baseline using the pixel-wise L_1 and SSIM metrics. For both, we considered three versions (traditional, inner and outer averages) in order to evaluate the effectiveness of the methods in different parts of the images. Our results suggested that the pyramidal strategy achieved superior results with the inner metrics, whereas the outpainting tended to perform better at the boundaries. ParLSI achieved the best results in most cases, especially with SSIM, but, in our visual inspection, the SeqLSI generated images with superior quality. We observed some artifacts in the outpainting-based versions that can be avoided using a more advanced network in the future.

References

1. Chen, X., Chen, X., Zha, Z.J.: Structure-Aware Residual Pyramid Network for Monocular Depth Estimation. In: International Joint Conference on Artificial Intelligence (2019)
2. Dhamo, H., Navab, N., Tombari, F.: Object-Driven Multi-Layer Scene Decomposition From a Single Image. In: IEEE/CVF International Conference on Computer Vision (2019)
3. Dhamo, H., Tateno, K., Laina, I., Navab, N., Tombari, F.: Peeking Behind Objects: Layered Depth Prediction from a Single Image. Pattern Recognition Letters (2019)
4. Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: DeepView: View Synthesis with Learned Gradient Descent. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)

5. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision Meets Robotics: The KITTI Dataset. International Journal of Robotics Research (2013)
6. Han, D., Kim, J., Kim, J.: Deep Pyramidal Residual Networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
7. Hoorick, B.V.: Image Outpainting and Harmonization using GANs - PyTorch Implementation (2020), https://github.com/basilevh/image-outpainting
8. Layton, O.W., Fajen, B.R.: Computational Mechanisms for Perceptual Stability using Disparity and Motion Parallax. Journal of Neuroscience (2020)
9. Liu, S., Li, T., Chen, W., Li, H.: Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In: IEEE International Conference on Computer Vision (2019)
10. Luvizon, D.C., Carvalho, G.S.P., dos Santos, A.A., Conceicao, J.S., Flores-Campana, J.L., Decker, L.G., Souza, M.R., Pedrini, H., Joia, A., Penatti, O.A.: Adaptive Multiplane Image Generation from a Single Internet Picture. In: Winter Conference on Applications of Computer Vision (2021)
11. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
12. Pinto, A., Córdova, M.A., Decker, L.G.L., Flores-Campana, J.L., Souza, M.R., dos Santos, A.A., Conceição, J.S., Gagliardi, H.F., Luvizon, D.C., Torres, R.S., Pedrini, H.: Parallax Motion Effect Generation Through Instance Segmentation And Depth Estimation. In: International Conference on Image Processing. IEEE (2020)
13. Shade, J., Gortler, S., He, L.w., Szeliski, R.: Layered Depth Images. In: 25th Annual Conference on Computer Graphics and Interactive Techniques (1998)
14. Shih, M.L., Su, S.Y., Kopf, J., Huang, J.B.: 3D Photography using Context-aware Layered Depth Inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition (2020)
15. Srinivasan, P.P., Tucker, R., Barron, J.T., Ramamoorthi, R., Ng, R., Snavely, N.: Pushing the Boundaries of View Extrapolation with Multiplane Images. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
16. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for Optical Flow using Pyramid, Warping, and Cost Volume. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
17. Tucker, R., Snavely, N.: Single-view View Synthesis with Multiplane Images. In: IEEE Conference on Computer Vision and Pattern Recognition (2020)
18. Tulsiani, S., Tucker, R., Snavely, N.: Layer-Structured 3D Scene Inference via View Synthesis. In: European Conference on Computer Vision (2018)
19. Van Hoorick, B.: Image Outpainting and Harmonization using Generative Adversarial Networks. arXiv preprint arXiv:1912.10960 (2019)
20. Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: Synsin: End-to-end View Synthesis from a Single Image. In: Conference on Computer Vision and Pattern Recognition (2020)
21. Zhang, M., Zhang, Y., Piao, Y., Liu, J., Ji, X., Zhang, Y.: Parallax based Motion Estimation in Integral Imaging. In: Digital Holography and Three-Dimensional Imaging. Optical Society of America (2019)
22. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo Magnification: Learning View Synthesis Using Multiplane Images. ACM Transactions on Graphics (2018)