3D Neural Style Transfer

Zoe Ghiron, Arec Jamgochian, and Bernard Lange
Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305
{zghiron, arec, blange}@stanford.edu

Abstract

Increased adoption of depth-estimating cameras renders useful the depth-aware transfer of artistic styles to content images. In this paper, we explore extensions of neural style transfer to three dimensions using iterative style transfer. For content image data that includes a processed depth channel, we implement alternative content and style loss functions that use depth to mask parts of the image. For image data that lacks a depth channel, we explore using a depth prediction network and a depth-matching loss component to mirror depth estimates between content and pastiche images. We investigate transferring single artistic styles, as well blending multiple. Ultimately, we showcase different methods for using depth to augment neural style transfer.

1. Introduction

In the last few years, we have witnessed an explosion in the number of research papers and web/phone applications (e.g. Prisma) performing style transfer techniques on photos. Typically in style transfer, a provided *content* image is modified to exhibit salient features of a *style* image, without affecting the nature of the content of the original image. The resultant *pastiche* image typically incorporates the style of the *style* image with the content of the *content* image.

Techniques from deep learning have been applied to tractably and efficiently perform 'neural' style transfer, but those techniques have been typically applied to 3-channel (RGB) images. With the rising popularity of dual-camera phones and other sensors capable of estimating depth, we expect three-dimensional neural style transfer to be an appropriate extension of two-dimensional style transfer's beauty. In this paper, we investigate applications of depth-aware neural style transfer to images where depth is either available as a fourth channel, or estimated via deep learning.

Specifically, in this project we formulate depth-dependent style transfer, by:

1. Implementing a depth-based mask to the style and content loss functions typically used in neural style transfer
2. Implementing a depth estimation network and loss function which tries to match the depth of the pastiche with the estimated depth of the content image.

With these adjusted and introduced losses, we can use depth information to create depth-dependent blending effects (e.g. stronger blending in the background of an image).

In task 1, we input single RGB-D content images obtained from the NYU Depth v2 Dataset [10] along with either single or multiple artistic style images and use an iterative loss-minimization approach [2] to output single, 3-channel (RGB) stylized images, where the magnitude of stylization is a function of depth.

In task 2, we input single RGB content images and run them through a depth prediction network prior to optimization. When stylizing our pastiche, we run it through the same depth prediction network and try to minimize the differences between estimated depths of our pastiche and our content image, while trading off between unmasked content and style losses. Again, our output is a single, 3-channel (RGB) stylized image.

2. Related Work

The algorithm for iterative neural style transfer is introduced in [2]. The main challenge of style transfer - the separation of an image content from the style - was addressed by using image representations acquired from the pre-trained Convolutional Neural Networks (CNN), i.e. 16-layer VGGNet [12]. The utility of the CNN is its capability to extract high-level image information through its feature representation. The authors show that one can can extract feature representation of both content and style images from the pre-trained CNN, and generate a new image by minimizing a loss function which balances between content and style (formulated in Section 3.1).

In iterative, or optimization-based style transfer, this is done by iteratively modifying an originally white-noise input through backpropagation of an appropriate loss function. This enables the generation of a new image (trained

on our white noise input instead of the architecture's parameters) with the desired style and content (i.e. content reconstruction) [9, 2].

Guided style transfer is explored in [3]. The authors show that you can guide the areas of stylization in the pastiche by applying a masking guidance channels T_l^r to the image, which they specify can be "either a binary mask for hard guidance or real-valued for soft guidance". In Section 3.2 we formulate a depth-dependent mask for soft guidance on both our content and style losses.

Depth-aware style transfer is explored in [8]. The authors explore implementing a depth prediction network on both the content image and the stylized pastiche, and adding a loss term that tries to match the two. Doing so ensures that the pastiche image still portrays salient landmarks at their appropriate depths, and that the stylization of the content image does not tarnish the three-dimensional content aspects. Self-supervised monocular depth estimation has been implemented via a feed-forward network. The `Monodepth2` feed-forward depth prediction network and has recently enjoyed high qualitative and quantitative success [5].

An alternative to iterative style transfer is network-based style transfer, in which one trains a style-specific feed-forward network to perform style transfer on a content image in a single forward pass [7, 13]. While training this feed-forward network is slow, it can be sped up by using instance normalization layers [14]. This has been further extended to train a feed-forward network for image generation with any arbitrary style [4]. Image-to-image translation using generative adversarial networks (`CycleGAN`) has also become a popular approach to style transfer [15]. Due to their training complexity, we do not explore these methods in this paper.

3. Methods

3.1. Naive Style Transfer

Naive iterative style transfer is performed by initializing a white-noise pastiche image and running gradient-based optimization on the pixels of the pastiche image in order to minimize a multi-objective loss function. This method relies on a pre-trained object detection network to propagate images through. Using such a pre-trained networks allows us to extract and compare high-level content and style image information. During the optimization, the weights in the feed-forward net remain fixed - only the pixel values in the pastiche image change.

In order to capture features of the content image and the style of style image, it is useful to define a loss function which trades off between the two [2]:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}.$$

In order to enforce smoothness in the pastiche image, we add on a regularization term to reduce pixel-to-pixel variation:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style} + \gamma\mathcal{L}_{tv}.$$

To represent the content of the image, we define $F^l \in \mathbb{R}^{N_l \times M_l}$, a feature response in layer l where N_l is the number of filters and M_l the flattened feature representation shape ($height \times width$). This enables us to set up a content loss which compares feature representation from the generated image (F^l) and real image (P^l, which has shape equivalent to F^l).

$$\mathcal{L}_{content} = \frac{1}{2}\sum_{i,j}(F_{i,j}^l - P_{i,j}^l)^2$$

The style is represented through the Gram matrix $G^l \in \mathbb{R}^{N_l \times N_l}$, which describes the feature correlations of different filter responses. This allows us to represent the texture information without the spatial content arrangement [2]. We define single layer contribution and total style loss below:

$$\mathcal{L}_{style} = \sum_{l=0}^{L}\omega_l E_l$$

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G_{ij}^l - A_{ij}^l)^2.$$

where ω_l is the weighting factors, E_l is layer loss contribution, G^l is the Gram matrix of our pastiche and A^l is its equivalent for our style image.

We now select layers at which to compare style and content. Based on [2], style loss should include losses distributed across layers to maintain the structure of the style. However, for the content loss only a single layer should be considered. In order to make our pastiche appear similar to the content image, it is beneficial to pick a lower layer to compare content. To make it appear more artistic, it is beneficial to pick a higher layer, as it represents a high-level feature representation of the content of the image without specific pixel to pixel matching. The approach is well represented in the Fig. 2 in [2].

To enforce smoothness, we define the regularizing pixel-to-pixel variational loss as

$$\mathcal{L}_{tv} = \sum_{c=1,i=1,j=1}^{3,H-1,W}(x_{i+1,j,c} - x_{i,j,c})^2 + \sum_{c=1,i=1,j=1}^{3,H,W-1}(x_{i,j+1,c} - x_{i,j,c})^2.$$

Once we choose appropriate relative weights and layers, we can perform optimization on the pastiche image while keeping the network weights fixed.

3.2. Masked Content and Style Losses

To extend style transfer to the depth dimension, we formulate ways in which we can use depth information in order to guide style transfer. Namely, we take inspiration from [3] to formulate both masked content losses and masked style losses.

We can adjust our content loss function by forcing some guided regions of the pastiche to match more closely to the content image compared to other regions:

$$\mathcal{L}_{content} = \frac{1}{2}\sum_{i,j}(\gamma_c(d^l_{i,j}) \odot (F^l_{i,j} - P^l_{i,j}))^2.$$

Here we can use γ_c to apply an element-wise function to our depth map at layer l in order to make appropriate behavior occur. An example of a useful functions γ_c might apply a linear transformation to transition depth to lie between 0.8 and 1.0 instead of 0.0 and 1.0 (since a depth of 0.0 would produce an image with no content weighting a particular region).

We can formulate masked style loss in a similar fashion, this time in the Gram matrix formulation. The matrix G^l is modified accordingly to include depth effects, where γ_s is a function of depth.

$$G^l_{ij} = \frac{1}{\eta^l}\sum_k(\gamma_s(d^l_{ik}) \odot F^l_{ik})(\gamma_s(d^l_{jk}) \odot F^l_{jk})$$

$$\eta^l = \overline{\gamma_s(d^l)}^2,$$

where $\overline{\gamma_s}$ indicates the mean of the matrix $\gamma_s(d^l)$, a necessary scaling factor since we do not adjust the Gram matrices of our style image. Note that again, γ_s enables us to control which parts of the image are targeted with the specific style. When it is set uniformly to one, we obtain our original Gram matrix formulation. An example of a useful function for guided style transfer is the sigmoid function, σ. Using $\gamma_s(d) = \sigma(20(d - 0.5))$ would cleanly separate the background from the foreground.

Once we have formulated these guided losses, we may use the same iterative style transfer methods described in Section 3.1. Results are shown in Section 5.2.

3.3. Depth Loss

A different expansion to the method introduced in section 3.1 was introduced by [8] . Apart from the content and style losses, additional depth component was introduced to the total loss. The depth loss and total loss are shown in the equation below. The objective of the added depth loss is to acquire a pastiche similar in style and content but also with a similar portrayed depth as the content image.

$$\mathcal{L}_{depth} = \frac{1}{HW}\left\|\phi(I_{original}) - \phi(I_{pastiche})\right\|^2$$

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style} + \gamma\mathcal{L}_{depth}$$

Though we assume a known depth channel in Section 3.2, it is still important to have a method of evaluating the depth in the pastiche image. A depth prediction network with a per-pixel minimum reprojection loss and masked photometric loss is presented in [5]. We use their depth prediction implementation [1] to estimate depth for 3-channel content images which lack a depth field.

4. Datasets and Networks

For the methods presented in 3.2, we use the NYU Depth v2 dataset [10], which includes a significant number of RGB-D images we can stylize. The dataset includes 1449 densely labeled pairs of 480x640 pixel RGB and Depth images recorded by a Microsoft Kinect.

We use a pre-processing script to filter raw depth data and download the dataset into a usable format. In order to have depth data be usable as a mask at different layers, we first normalize the filtered depth data on a zero-one scale. We additionally normalize pixel data to a zero-one scale. To portray depths accurately at each layer which we calculate losses, we interpolate our depth data to be of same height and width of convoluted image data at that layer.

For the method presented in Section 3.3, we don't need RGB-D data as depth estimation is done by the neural network. Hence two images shown in Fig. 9 and Fig. 10 are considered. The second image is an example of ideal depth prediction taken from [5].

For style data, we use selected artworks that are typically benchmarked with. Examples include Van Goghs *'A Starry Night'*, Kandinsky's *'Composition VII'*, and Munch's *'The Scream'*.

For our feed-forward convolutional neural network for capturing image features, we are using Squeeze Network [6] and VGG16 [12], with weights trained on the ImageNet training dataset. The SqueezeNet is useful for the method described in Section 3.2 because of its small size and low number of parameters, making for quick optimization. For the method described in Section 3.3 we consider VGG16 as it provides additional detail in the pastiche image and make it easier to highlight differences caused by depth component of the loss. For the depth estimation we use implementation done in [1], which uses an adapted ResNet architecture described in [5].

5. Results

We implemented the aforementioned methods using PyTorch [11]. Most of the parameters involved in the style transfer optimization must be tuned image-wise to achieve

[1] available at https://github.com/nianticlabs/monodepth2

satisfying results. Adam was used to optimize the loss because it speeds up optimization. For content loss, we use one of the lower layers to capture realistic content. Several layers across the network are used for the style loss.

5.1. Naive Style Transfer

For baseline results, we have implemented 2D style transfer via optimization of a randomly initialized image to trade off between the aforementioned content and style losses, including a variational regularizer to reward continuity in the image. Results can be seen in Fig. 1.

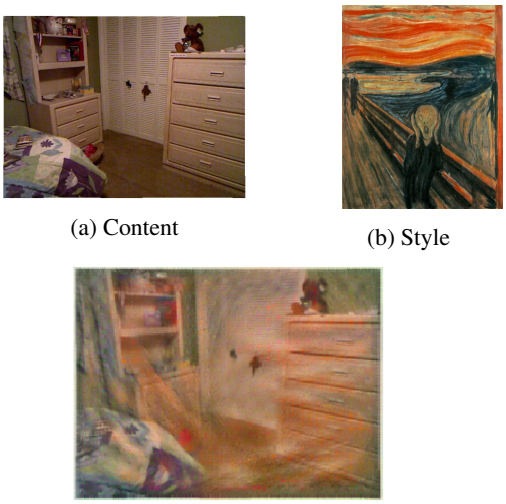
Figure 1: Content Image + Style Image = Pastiche Image with Vanilla Style Transfer.

We also chose a content image from the NYU Depth v2 dataset (Fig. 2a), with *"The Scream"* for the style (Fig.). The pastiche in Fig. 2c was produced optimizing a naive style transfer loss, as a basis for further quantitative and qualitative comparison with 3D-style transfer. The loss converged to 3195 in $29.53s$ on a CPU, in 300 iterations (learning rate is decayed from 3 to 0.1 at iteration 180).

5.2. Masked Content and Style Losses

We implemented the 3D style transfer, masking the content or the style loss with a function of the depth channel of the input RGB-D image. The resulting depth mask for the presented example is shown in Fig. 3.

The pastiche images generated with a style mask and with a content mask are shown in Fig. 4 and Fig. 5 respectively. If we compare the pastiche generated without mask (Fig. 2c) and the pastiche generated with the style mask (Fig. 4), it is clear that no style is applied at the background of the picture, were the depth mask is dark (see Fig. 3). Similarly, the background content has disappeared on the pastiche generated with the content mask (Fig. 5). From these observations, we can conclude that applying a mask to content or style hides the chosen feature from the depth-selected place. Still, using the content mask emphasizes style where the content is masked, and using the style mask emphasizes content were the style is masked. Playing



(a) Content

(b) Style

(c) Pastiche image

Figure 2: RGB-D content + style = Pastiche image (vanilla style transfer)

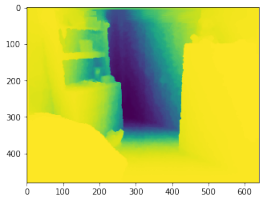


Figure 3: Processed depth, $\gamma(d) = 1 - \sigma(20(d - 0.8))$

Figure 4: Style transfer with a style mask

around with both of the masks and their parameters, we are able to generate a wide range of depth-dependent mixtures of style and content.

For the style mask, the loss converges in 300 iterations to 3353. For the content mask, it converges in 300 iterations to 2911. As a reminder, the basic 2D style transfer

Figure 5: Style transfer with a content mask

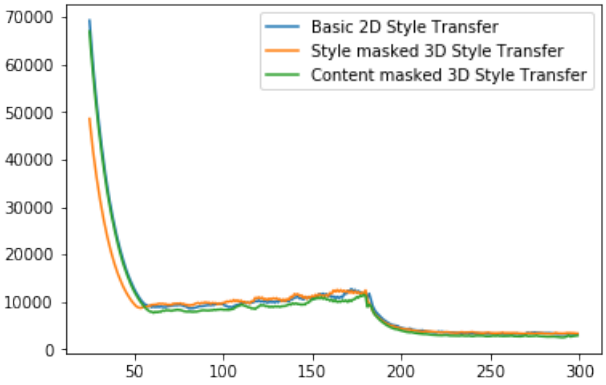
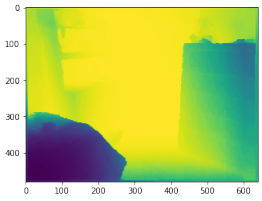
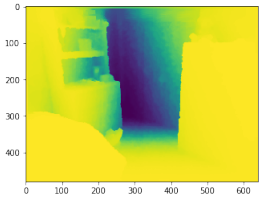


Figure 6: Evolution of the loss with iterations for basic 2D style tranfer, style masked and content masked 3D style transfer



(a) 1st style: depth mask

(b) 1st style



(c) 2nd style: depth mask

(d) 2nd style

Figure 7: Pastiche of 3D-style transfer with two styles

converged to a cost of 3195 in 300 iterations. Each of these convergences have been established for the same set of hyperparameters.

In Fig. 6, we notice that the three examples converge in 300 iterations. In addition, it is clear that the evolution of the cost has the same shape for the 2D and 3D style transfer.

The style loss function is in average 1.32% longer to compute (on a CPU) if the style must be masked. Similarly, the content loss function is in average 1.68% longer to compute (on a CPU) if the content must be masked. The average was computed over 300 samples.

This result is satisfying considering that these functions are computed 300 times over the optimization of the loss which takes approximately $40s$ on the CPU.

Finally, we demonstrated the ability to perform style transfer with two different styles, the blending of the two styles being dependent on the depth. This was performed adding a second style loss function to the loss. The first style loss is computed based on the first style (Fig. 7b) and the first depth mask (Fig. 7a). The second style loss is computed based on the second style (Fig. 7d) with the second depth mask (Fig. 7c). The resulting pastiche is shown in Fig. 7. It is clear that the first style is applied to the background of the input image when the second style is applied to the foreground. This demonstrates the ability to control style transfer using depth.

5.3. Depth Loss

The baseline for the style transfer with depth loss is the naive style transfer method implemented with VGG16 architecture [12]. Two example images are presented in Fig. 9a and 10a with their respective naive pastiche images (Fig. 9c and 10c). To increase the effectiveness of the style transfer, the content layer is one of the last (28th). Decrease in the content layer number, make the pastiche look closer to the content image. The output of the depth prediction network is shown in Fig. 9b and 10b [1]. As can be seen, in Fig. 9 depth estimation performs acceptably well, whereas Fig. 10's depth map is much closer to reality. Fig. 10 from [1] represents the ideal scenario for the depth prediction

network. The pastiche images with depth loss component are shown in Fig. 9d and 10d. Based on the outputs, style transfer with depth loss definitely maintains the content and the style of the image acceptably when tuned appropriately. Apart from the image saturation which we consider random (it is heavily affected by the numerous parameters which are difficult to analyze at the early stage of parameter tuning), the main difference is the sharpness of the image depending on the depth. It can be seen that the parts of the image with shallower depth are significantly sharper than the one further, e.g. in Fig. 9, the sky and the tower appear blurry whereas the bus's sharpness remains comparable to the naive implementation.

Moreover, apart from the sharpness variation depending on the depth map, characteristics of the objects with low depth (i.e. shallow depth) are heavily altered. When we look at the front wheel and the bonnet of the car in the Fig. 10, we can notice that it remains well reconstructed apart from the parts which are not visible in the depth map, e.g. in the depth map, the wheel is flush with the bodywork of the vehicle whereas in the original image and naive style transfer pastiche is not. In the style transfer with depth component, wheel is still there thanks to the content loss component but it is flush with the bodywork of the car to align with the depth map. It implies that with current parameters, the depth map is prioritized over the visual shape of the image purely based on the content feature representation. The same trend is visible in the reconstruction of the curve of the bonnet of the car.

As can be seen in the Fig. 8, the loss convergence for the style transfer with depth component converges to the value of approx. 10000 (heavily depends on the hyperparameters) but it is heavily unstable. It is challenging to optimize the loss with extra depth component. In this method, on top of the forward and backward pass through the feature representation neural network (in this case VGG16), we add a pass through depth prediction network based on ResNet at each optimization step. It makes it infeasible to run solely on CPU. What is more, "convergence" to acceptably visually image took significantly more iterations which forced us to use a GPU, Nvidia Tesla K80 and P100 were used. The benefit of switching from CPU to GPU in this scenario is clear. The boost of 315%s in the compute speed of one iteration (approx. 3.8s with CPU per iteration). For comparison, one iteration of naive style transfer and the style transfer with depth component run on GPU take 0.016s and 1.21s, respectively, which makes it approx. 75 times longer to compute per iteration. The parameters used for the images generated in the Fig. 9 and 10 are: image and style size: 192, content layer: 28, content weight: 6e-2, style layers: multiple layers with weights decreasing along the layer number, tv weight: 2e-2 and depth weight: 10000. The depth weight was increased to verify its influence on

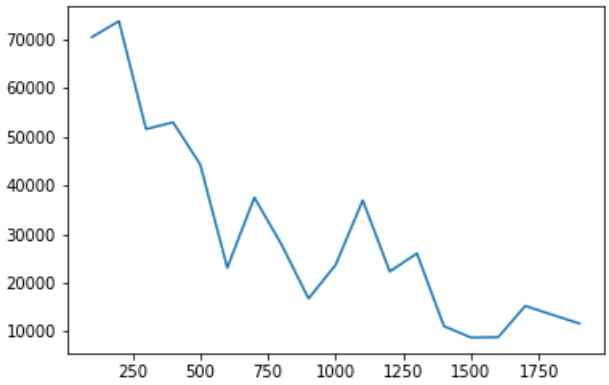


Figure 8: Evolution of the loss with iterations for basic 2D style tranfer, style masked and content masked 3D style transfer

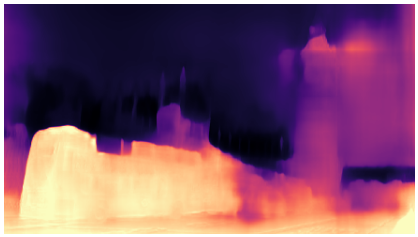
the image creation (between 1 and 200000). However, the complexity of each optimization step and the number of iterations make the extensive analysis impossible as a course project. Hence, further hyper-parameters optimization is a necessity as a future work.

6. Conclusion and Future Work

The main objective of the project was to explore different methods to perform depth-dependent style transfer.
The first elaborated technique was to mask the pastiche image with a depth-dependent mask during training in order to "hide" some regions to the style or to the content loss. As a result, the mix of style and content will differ depending on the depth. This technique is efficient: its convergence time is similar to basic 2D iterative style transfer and the output pastiches are satisfying. However, it is reliant on RGB-D images. If we were to continue this project in the future, we could incorporate a depth estimator as a pre-processor that would estimate the depth map for a basic RGB image. Then, the 3D style transfer could be applied to any kind of images.

In the second technique, we investigated adding a depth loss component to the basic style transfer loss which ensures that the estimated depth of the pastiche matches pixel-wise with the estimated depth of the content image. This method doesn't rely on implicit depth data, so it can be applied to any image. The downside of this method is its computational efficiency: depth estimation must be computed at each iteration and the integration of the two networks (loss optimization and depth estimation) is not well optimized. In the future, it would be exciting explore more efficient incorporation of depth prediction network. Further, hyper-

(a) Original Image



(b) Depth Map

(c) Naive style transfer

(d) Style transfer with depth component

Figure 9: Example image

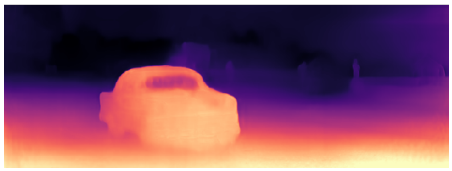
parameter tuning is definitely a necessity to perform style transfer at high resolution images.

Contributions & Acknowledgements

The style transfer notebook from the third CS231N homework assignment was used as a foundation to implement the optimization of the masked content and style losses. The depth prediction network used the public code [1].

Arec elaborated the literature review for the project, formulated the masked content and loss functions, extracted the NYU Depth v2 dataset, and structured the code. Zoe coded the masked losses and tuned the hyperparameters to produce pastiches. Bernard worked on the use of VGG16, loss functions primarily focusing on the implementation of

(a) Original Image



(b) Depth Map

(c) Naive style transfer

(d) Style transfer with depth component

Figure 10: Example image from [5] with its depth map, naive style transfer and style transfer with depth component

depth loss component and integration of monodepth2 implementation [1].

We also acknowledge Justin Johnson and our project milestone reviewer for their recommendations.

References

[1] https://github.com/nianticlabs/monodepth2. 3, 5, 7

[2] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 1, 2

[3] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3985–3993, 2017. 2, 3

[4] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017. 2

[5] C. Godard, O. Mac Aodha, and G. J. Brostow. Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260, 2018. 2, 3, 7

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 3

[7] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. 2

[8] X.-C. Liu, M.-M. Cheng, Y.-K. Lai, and P. L. Rosin. Depth-aware neural style transfer. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, page 4. ACM, 2017. 2, 3

[9] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 2

[10] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1, 3

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 3

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 3, 5

[13] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016. 2

[14] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2

[15] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2