Laplacian-Steered Neural Style Transfer

Shaohua Li[1,2] shaohua@gmail.com
Xinxing Xu[2] xuxinx@ihpc.a-star.edu.sg
Liqiang Nie[3] nieliqiang@gmail.com
Tat-Seng Chua[1] dcscts@nus.edu.sg
1. School of Computing, National University of Singapore
2. Institute of High Performance Computing, Singapore 3. Shandong University

ABSTRACT

Neural Style Transfer based on Convolutional Neural Networks (CNN) aims to synthesize a new image that retains the high-level structure of a content image, rendered in the low-level texture of a style image. This is achieved by constraining the new image to have high-level CNN features similar to the content image, and lower-level CNN features similar to the style image. However in the traditional optimization objective, low-level features of the content image are absent, and the low-level features of the style image dominate the low-level detail structures of the new image. Hence in the synthesized image, many details of the content image are lost, and a lot of inconsistent and unpleasing artifacts appear. As a remedy, we propose to steer image synthesis with a novel loss function: the Laplacian loss. The Laplacian matrix ("Laplacian" in short), produced by a Laplacian operator, is widely used in computer vision to detect edges and contours. The Laplacian loss measures the difference of the Laplacians, and correspondingly the difference of the detail structures, between the content image and a new image. It is flexible and compatible with the traditional style transfer constraints. By incorporating the Laplacian loss, we obtain a new optimization objective for neural style transfer named Lapstyle. Minimizing this objective will produce a stylized image that better preserves the detail structures of the content image and eliminates the artifacts. Experiments show that Lapstyle produces more appealing stylized images with less artifacts, without compromising their "stylishness".

KEYWORDS

Neural Style Transfer, Convolutional Neural Networks, Image Laplacian.

1 INTRODUCTION

Neural Style Transfer based on Convolutional Neural Networks (CNNs) [6, 7, 11, 17] has gained a lot of attention from both academia and industry. Given two input images, a content image and a style image, the Neural Style Transfer algorithm synthesizes a new image, referred to as the *stylized image*, which retains the basic structure and semantic content of the content image, but is rendered in similar texture as the style image. Fig. 1 presents an example of style transfer: given a content image 1(a) and a style image 1(b), a stylized image 1(c) is synthesized. This combination is achieved thanks to the factorized representations of an image by CNNs: the high-level CNN features compactly encode the semantic content, e.g. the objects and global structure of the input image, whereas the low-level features encode the local details, e.g. colors, basic shapes and texture. Neural Style Transfer specifies a total loss for a new image, which consists of both the content loss, i.e. its high-level feature difference with the content image, and the style loss, i.e. its low-level feature difference with the style image. An image minimizing the total loss will combine the global semantic content of the content image and the local texture of the style image. One limitation of the current methods is that, they do not incorporate the local features of the content image into the optimization objective. Consequently, the *detail structures* (edges, contour segments, patterns, gradual color transitions, etc.) of the stylized image are dominated by the low-level style image features. After the optimization, visual elements in the style image may be inappropriately transferred to random positions in the stylized image, forming unappealing irregular artifacts, and some detail structures in the content image may be severely distorted. These artifacts and distortions are especially prominent on regions such as faces or object contours, to which human perception are sensitive (e.g. Fig. 1c). For notational convenience, the traditional method by Gatys et al. [6] is referred to as *Gatys-style*.

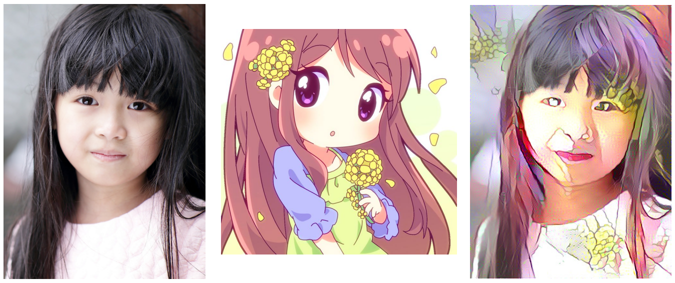
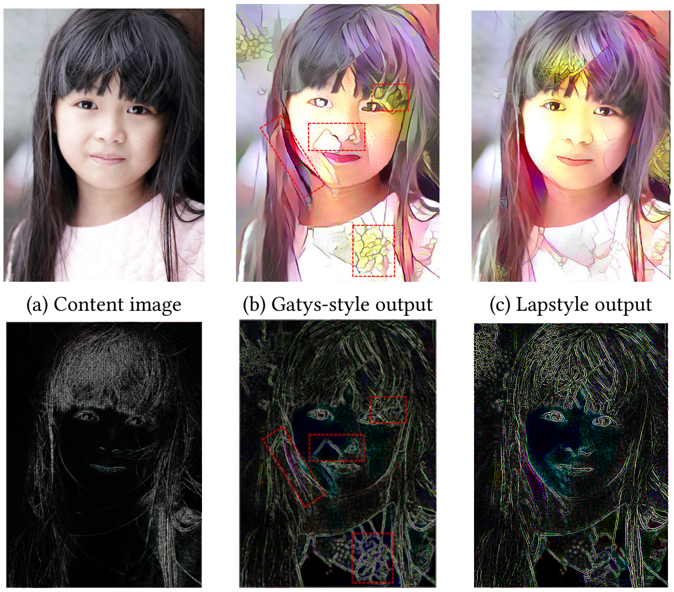

(a) Content image (b) Style image (c) Gatys-style

Figure 1: Given a content image (a) and a style image (b), style transfer creates a new one combining the content of (a) and the style of (b). (c): the stylized image by Gatys et al. [6] contains a lot of unpleasing artifacts, such as those on the girl's face.

Naturally, to improve the structural coherence of neural style transfer, more constraints with respect to the content detail structures could be incorporated. When adding new content constraints, however, one need to be cautious not to disrupt the optimization of the style loss. A rigid content constraint will greatly reduce the search space for feasible solutions, where a solution with a small

MM '17, October 23–27, 2017, Mountain View, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4906-2/17/10…$15.00
https://doi.org/10.1145/3123266.3123425


(a) Content image (b) Gatys-style output (c) Lapstyle output

(d) Content Laplacian (e) Gatys-style Laplacian (f) Lapstyle Laplacian

Figure 2: Correspondences between Laplacian deviations and image distortions. (a), (d): the content image and its Laplacian. (b), (e): the stylized image by Gatys-style and its Laplacian. Major Laplacian deviations in (e) are highlighted; the corresponding regions in (b) are severely distorted. (c), (f): the stylized image by Lapstyle and its Laplacian. Due to the Laplacian loss term, the major Laplacian deviations are alleviated in (f), and the corresponding regions in (c) become more faithful.

style loss may not exist. For example, in Section 4, we empirically show that a constraint w.r.t. some low-level CNN features of the content image is too rigid and greatly reduces the "stylishness" of the result image.

In the domain of computer vision, the Laplacian operator is widely used to detect edges (regions of rapid intensity changes) [1, 24]. The Laplacian operator extracts a Laplacian matrix (a *Laplacian* in short) that contains the second-order variations in an image that are prominent to human perception [15, 22]. These second-order variations often correspond to detail structures. As an example, Fig. 2(d) is the Laplacian obtained from the previous content image, which makes explicit important edges and boundaries within the content image. Large deviations in the Laplacian usually correspond to large distortions or artifacts in the stylized image. For example, the highlighted regions in Fig. 2(e) correspond to distorted regions in Fig. 2(b). Naturally, by imposing a new constraint that requires the stylized image to have a similar Laplacian to the content image, the stylized image may better preserve the detail structures.

An image Laplacian constraint has the advantage that, since it is applied to the second derivatives of image pixels, many images correspond to the same Laplacian, and each of the image is a solution of this Laplacian constraint. After putting other constraints back, an image well satisfying all constraints may still exist. Poisson Image Editing [22] is an example of incorporating a Laplacian constrant with some boundary constraints to transfer a region seamlessly. In other words, a Laplacian constraint is flexible and compatible with other constraints, e.g. the aforementioned content and style feature constraints. Motivated by these virtues of the Laplacian, we propose to add a *Laplacian loss* into neural style transfer, to steer the stylized image towards having a similar Laplacian to that of the content image. The Laplacian loss is defined as the mean-squared distance between the two Laplacians. Minimizing this loss drives the stylized image to have similar detail structures as the content image. Meanwhile, the stylized image will still be rendered in the new style. This enhanced neural style transfer method is named **Lapstyle**. Fig. 2(c) show that in the presence of the Laplacian loss, severe Laplacian deviations are eliminated or alleviated, and the stylized image is more faithful to the original content image.

The Laplacian loss is computed by a small two-layer fixed CNN. This network consists of an average pooling layer and a prespecified convolutional layer, and can be easily plugged into almost all existing style transfer methods to better preserve the detail structures of the content image. The computational overhead of the Laplacian loss to an existing method is negligible. Moreover, we have released our source code of Lapstayle[1] to facilitate further research.

We evaluated Lapstyle on various test images under different conditions. Experimental results show that compared to the popular Gatys-style method [6], Lapstyle always produces stylized images more faithful to the content images in low-level structures, with much less artifacts and distortions, and almost equally "stylish". In addition, we measured how the presence of the Laplacian loss term impacts all types of losses in the optimization objective, and confirmed that the Laplacian loss term drastically reduces the Laplacian loss between the content and stylized images, at the cost of only slight increases of the content and style losses. This supports the above empirical observations quantitatively.

A natural extension to the Laplacian loss is to combine multiple losses with pooling layers of different sizes, to capture different scales of the image detail structures. We investigated several combinations of two Laplacian losses, and presented an example improved by the Laplacian combinations.

To sum up, the contributions of this work are threefold:

(1) We propose a novel detail-preserving loss named *Laplacian loss* for neural style transfer, which steers the synthesized image towards having similar low-level structures as the content image, while being flexible to allow the image to be rendered in the new style. Augmented by the Laplacian loss, a new style transfer method named Lapstyle is obtained. Extensive experiments show that Lapstyle generates stylized images that are more appealing with less artifacts.
(2) The Laplacian loss is computed through a small add-on CNN. This loss can be easily incorporated into almost any existing style transfer methods for better preserving the detail structures of the content image.
(3) Multiple Laplacian losses with different granularities can be combined to capture the image detail structures at different scales. We have empirically investigated several combinations and observed improvements on some examples.

[1] https://github.com/askerlee/lapstyle

2 RELATED WORK

The task of style transfer has been studied in computer vision and graphics communities for a long time, under the name of "texture transfer". Some early works synthesize new images by sampling pixels or image patches according to certain similarity metrics [5, 13, 14, 25]. Another line of works first extract a set of feature maps from an image by applying manually designed filters, and then transform the feature maps of a synthesized image to match that of a style image [10, 23].

An important inspiration for the present work is Poisson Image Editing [22], which aims to transfer a source region in an image to a target region in another image seamlessly. Poisson Image Editing restrains the target regions to have the same Laplacian as the source region, and fixes the boundary pixels to be the original pixels around the target region. As the Laplacian constraint has a big set of solutions, given the additional boundary constraint, a solution still exists. This solution is a transferred region fitted in the background in the target image with smooth transitions.

Agarwala et al. [2] and Darabi et al. [4] incorporate gradient terms into the patch similarity metric for matching structurally coherent patches and blending a region into another image with higher consistency. Recently, Lee et al. [15] propose to complete missing regions in an image with a Laplacian pyramid, i.e. stacked Laplacian filters in decreasing granularities. This method selects patches that are structurally coherent by exploiting the property that image Laplacians preserve the detail structures.

All the above methods are traditional style transfer methods, as they either operate at the pixel level, or use manully designed filters. Moreover, non-parametric, ad-hoc procedures are usually involved. Hence they can hardly capture and preserve high-level semantic information, and the manually designed components limit their generalization ability. As a result, the synthesized images are often globally incongruous or of low quality.

Gatys et al. [6] propose to use a convolutional neural network (CNN) to extract features for style transfer, namely the Neural Style Transfer method. Different layers of CNNs capture information at different levels of abstraction. Constraints with respect to different levels of features are integrated in a single CNN as different loss functions. In particular, the style to be transferred is captured as Gram correlation matrices of the style image features. In this framework, image synthesis reduces to the optimization within the CNN. This approach elegantly addresses the style transfer problem with high-quality output images. Gatys et al. [7] make two extensions to this method: 1) better control the spatial consistency by using a guided Gram loss, and 2) better preserve the color distribution of the content image, by incorporating an extra luminance channel or matching colour histograms.

Johnson et al. [11] propose a CNN for transforming images, named the transformation network. It is trained with the loss function as in [6], but the parameters to be trained are the CNN weights, instead of an input image. For each style image, a dedicated transformation network is trained. To get a stylized output image, one only needs to feed the content image into this CNN and take a forward pass, which is much faster than [6]. Li et al. [17] apply the traditional idea of patch-matching to CNN features. They propose a Markov Random Field loss function that drives the CNN feature patches in the synthesized image to approximate their best matches in the style image. This method, namely MRF-CNN, is extended by Champandard [3], in which user-provided segmentation information is incorporated to improve the local consistency of stylized images.

More recently, Luan et al. [18] propose a method that can achieve photorealistic style transfer, as a post-processing step of Neural Style Transfer. They first extract locally affine functions using Matting Laplacian[2] of Levin et al. [16], then fine-tune the stylized image to fit these affine functions. In the examples they presented, the stylized images are photorealistic and well retain the detail structures of the content image. On our test images (Section 4), this method produces images that are much more consistent to the content images, but their stylishness is greatly reduced.

3 METHOD

3.1 Neural Style Transfer

Given a content image $\boldsymbol{x}_c$ and a style image $\boldsymbol{x}_s$, let their corresponding CNN features at layer l be denoted as $F_l(\boldsymbol{x}_c)$ and $F_l(\boldsymbol{x}_s)$, respectively. Suppose there are N_l filters in layer l, then $F_l(\boldsymbol{x}) \in \Re^{M_l(x)\times N_l}$ is a matrix with N_l columns. Each column of $F_l(\boldsymbol{x})$ is a vectorized feature map, i.e. the response of a filter in layer l, containing $M_l(\boldsymbol{x}) = H_l(\boldsymbol{x}) \cdot W_l(\boldsymbol{x})$ elements, where $H_l(\boldsymbol{x})$ and $W_l(\boldsymbol{x})$ are the height and width of each feature map, respectively, depending on the size of the input image $\boldsymbol{x}$.

Neural style transfer generates an image $\hat{\boldsymbol{x}}$ that depicts the content of image $\boldsymbol{x}_c$ in the style of $\boldsymbol{x}_s$, by minimizing the following loss function of $\hat{\boldsymbol{x}}$ [6, 7]:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{content}} + \beta\mathcal{L}_{\text{style}}, \tag{1}$$

where the content loss $\mathcal{L}_{\text{content}}$ is the mean-squared distance between the feature maps of $\boldsymbol{x}_c$ and $\hat{\boldsymbol{x}}$ at a prespecified *content layer* l_c:

$$\mathcal{L}_{\text{content}} = \frac{1}{N_{l_c} M_{l_c}(\boldsymbol{x}_c)} \sum_{ij} (F_{l_c}(\hat{\boldsymbol{x}}) - F_{l_c}(\boldsymbol{x}_c))^2_{ij}, \tag{2}$$

and the style loss $\mathcal{L}_{\text{style}}$ measures the distributional difference of the feature maps of $\boldsymbol{x}_c$ and $\hat{\boldsymbol{x}}$ at several prespecified *style layers*:

$$\begin{aligned} \mathcal{L}_{\text{style}} &= \sum_l w_l E_l(\hat{\boldsymbol{x}}, \boldsymbol{x}_s) \\ E_l(\hat{\boldsymbol{x}}, \boldsymbol{x}_s) &= \frac{1}{4N_l^2} \sum_{ij} (G_l(\hat{\boldsymbol{x}}) - G_l(\boldsymbol{x}_s))^2_{ij}, \end{aligned} \tag{3}$$

where w_l is the weight of layer l, and $G_l(\boldsymbol{x}) = \frac{1}{M_l(\boldsymbol{x})} F_l(\boldsymbol{x})^\top F_l(\boldsymbol{x}) \in \Re^{N_l \times N_l}$ is the Gram matrix of the feature maps at layer l. The i, j-th element of $G_l(\boldsymbol{x})$ measures the correlation between the i-th and j-th feature maps, i.e. how often the i-th and j-th filters are simultaneously activated across small image patches.

The style loss on Gram matrices is essential to the impressive performance of Neural style transfer. The intuitions behind this formulation are as follows. A style or texture usually contains local and repetitive combinations of multiple visual elements. Suppose different visual elements activate different convolutional filters, then repetitive local co-occurrence of these elements correspond

[2] This 'Laplacian' refers to the graph laplacian, which is irrelevant to the image Laplacian used in this work.

to repetitive co-activation patterns in the feature maps, and consequently, yield large positive numbers in certain locations in the Gram matrix. By forcing the stylized image $\hat{\boldsymbol{x}}$ to have similar Gram matrices to $\boldsymbol{x}_s$, $\hat{\boldsymbol{x}}$ has to generate similar filter co-activation patterns, which probably indicates that $\hat{\boldsymbol{x}}$ possesses similar styles and textures. Different style layers are used to capture the styles and textures at different granularities.

Neural style transfer methods usually employ a pretrained 19-layer VGG network to extract features and perform the optimization, because VGG preserves more information at the convolutional layers [21]. In the original work [6], Gatys et al. chose 'conv4_2' as the content layer, and 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1' as the style layers.[3]

The optimization (image synthesis) proceeds as follows:

(1) Set the input of the CNN as the content image $\boldsymbol{x}_c$. Do a forward propagation and save the response $F_{l_c}(\boldsymbol{x}_c)$ of the content layer;

(2) Set the input of the CNN as the style image $\boldsymbol{x}_s$. Do a forward propagation. Compute and save the Gram matrices $\{G_l(\boldsymbol{x}_s)\}$ of all style layers;

(3) Initialize $\hat{\boldsymbol{x}}$. $\hat{\boldsymbol{x}}$ could be randomly initialized, or copied from the content image;

(4) Iterate until reaching N iterations:
    (a) Do a forward propagation, and compute the total loss;
    (b) Do a backward propagation to update $\hat{\boldsymbol{x}}$.

3.2 Laplacian Operator

The Laplacian operator $\triangle$ of a function f is the sum of all unmixed second partial derivatives:

$$\triangle f = \sum_i \frac{\partial^2 f}{\partial x_i^2}.$$

The Laplacian filter [24] is the discrete approximation to the two dimensional Laplacian operator, given by

$$D = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

The Laplacian matrix ("Laplacian" in short) of an image $\boldsymbol{x}$ is obtained by convolving the image with D, denoted by $D(\boldsymbol{x})$. At regions where adjacent pixel values change drastically, the convolution will produce a response of high magnitude, regardless of the direction of the change. In regions where the change is flat, the response is zero [1]. Hence the Laplacian operator is widely used for edge detection and extraction of detail structures in an image [15].

3.3 Laplacian Loss and Lapstyle Objective

Given two images $\boldsymbol{x}_c$ and $\hat{\boldsymbol{x}}$, we use a *Laplacian loss* $\mathcal{L}_{\text{lap}}$ to measure the difference between their Laplacians:

$$\mathcal{L}_{\text{lap}} = \sum_{ij} (D(\boldsymbol{x}_c) - D(\hat{\boldsymbol{x}}))^2_{ij}. \tag{4}$$

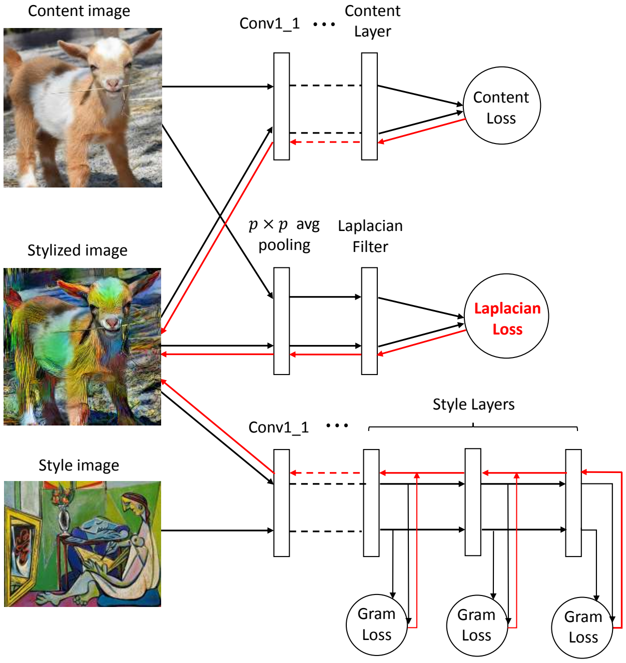


Figure 3: Network Architecture of Lapstyle. Black and red lines are forward and backward passes, respectively. Dashed lines indicate that there are unshown intermediate layers.

In order to force the stylized image to possess similar detail structures as the content image, we augument the style transfer objective (1) with the Laplacian loss $\mathcal{L}_{\text{lap}}$ of $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}_c$:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{content}} + \beta\mathcal{L}_{\text{style}} + \gamma\mathcal{L}_{\text{lap}}, \tag{5}$$

where γ is a tunable parameter controlling the strength of the Laplacian loss. In situations where the output image is severly distorted, we could increase γ to demand a more faithful stylization.

The new optimization objective (5) and the corresponding neural style transfer method is named *Lapstyle*. For the Lapstyle objective, the optimization procedure in Section 3.1 remains unchanged, except that in Step 1 the Laplacian $D(\boldsymbol{x}_c)$ needs to be saved. Fig. 3 presents the overall structure of Lapstyle.

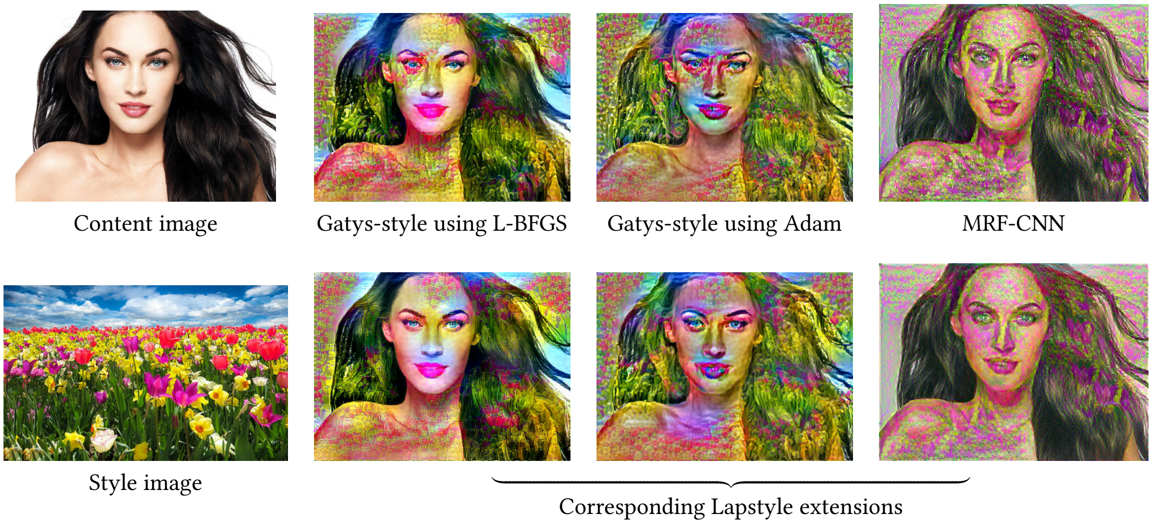
When computing the Laplacian loss, two practical issues are addressed as follows:

Laplacians on different channels The input image $\boldsymbol{x}$ has three RGB channels. Ideally, to detect edges, each channel should be considered separately, i.e. each channel convolves with the Laplacian filter, yielding three intermediate Laplacians. A value with a large magnitude in any of the three Laplacians suggests an edge, regardless of its sign. To this end, the final Laplacian $D(\boldsymbol{x})$ used in the Laplacian loss should be the sum of the absolute values of the three Laplacians [9]: $D(\boldsymbol{x}) = \left\|D(\boldsymbol{x}^R)\right\| + \left\|D(\boldsymbol{x}^G)\right\| + \left\|D(\boldsymbol{x}^B)\right\|$. However, this formulation involves some technical difficulties in implementation. For simplification, we adopt an approximation using their sum as the final Laplacian:

$$D(\boldsymbol{x}) = D(\boldsymbol{x}^R) + D(\boldsymbol{x}^G) + D(\boldsymbol{x}^B). \tag{6}$$

This summation is automatically calculated by convolving the three channels altogether with a Laplacian convolutional filter. We

[3] In their implementation, they actually used 'relux_y' in place of each 'convx_y', e.g. 'relu_4_2' instead of 'conv4_2', where negative responses are truncated to 0.



Content image

Gatys-style using L-BFGS

Gatys-style using Adam

MRF-CNN

Style image

Corresponding Lapstyle extensions

Figure 4: Style transfer results using differnt implementations (first row) and their corresponding Lapstyle extensions (second row). L-BFGS implementation of Gatys-style produces the best stylized image among the original implementations. Lapstyle extensions always produce better images than the original ones.

have not observed noticable degradation of image quality with this approximation.

Smoothing with a pooling layer The Laplacian filter is sensitive to small perturbations in the input image, and smoothing the input image can make the Laplacian loss better reflect its true detail structures. Hence we add a $p \times p$ average pooling layer before the Laplacian layer for smoothing. Pooling also reduces the memory overhead of the Laplacian loss to $1/p^2$ of that on the original image.

3.4 Combining Multiple Laplacians

When the pooling layer has a wider kernel, it condenses a larger area into one pixel. A Laplacian on top of such a pooling layer may capture structures in larger regions. Hence, combining multiple Laplacian losses over increasingly dilated pooling layers may capture detail structures in different granularities. The optimization objective is accordingly extended to

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{content}} + \beta \mathcal{L}_{\text{style}} + \sum_k \gamma_k \mathcal{L}_{\text{lap}k}, \tag{7}$$

where $\mathcal{L}_{\text{lap}k}$ is the Laplacian loss on the images pooled by an average pooling layer of size $p_k \times p_k$, and γ_k is its weight.

3.5 Implementation Details

The Laplacian filter is a 3x3 convolutional layer with fixed weights. Hence the Laplacian loss can be conveniently implemented in any mainstream deep learning frameworks and incorporated into most existing neural style transfer methods.

We have made extensions to three commonly used neural style transfer packages, and found that Gatys-style implemented using the L-BFGS optimization method yields the best stylized images. See Section 4 for more details.

4 EXPERIMENTS

In this section, we show the style transfer results on various content and style images, using the Lapstyle extensions to three commonly used neural style transfer packages. Compared to the the original packages, Lapstyle constantly yields more appealing images that better preserve the detail structures of the content image.

Baselines. Four commonly used neural style transfer packages are used as baseline methods. Among them, two are implementations of Gatys-style [6], one implemented by Justin Johnson[4], using the L-BFGS optimization method, and the other by Anish Athalye[5], using the Adam adaptive optimization method [12]. The third package is MRF-CNN [17] implemented by Alex J. Champandard[6]. The fourth package is the recent Photorealistic Style Transfer [18][7], referred to as Photo-style in the following.

Lapstyle Settings. Each of these three packages were extended with the Laplacian loss to get their Lapstyle counterparts. Due to the distinctiveness of each implementation, different sizes of the pooling layer and the Laplacian weight were adopted for different baselines. For the extension to the L-BFGS implementation of Gatys-style, we use a pooling size $p = 4$, the Laplacian weight $\gamma = 100$, and the total iterations $N = 1000$. For the extension to the Adam implementation of Gatys-style, we fix $p = 2, \gamma = 200, N = 1000$. For the extension to MRF-CNN, we set $p = 2, \gamma = 100, N = 300$. For Photo-style, we set $\lambda = 1000$.

Fig. 4 compares the stylized images using a same test pair. Clearly, the L-BFGS implementation of Gatys-style and its corresponding Lapstyle extension yields the best images. In fact, the L-BFGS implementation of Gatys-style has been observed to consistently perform the best on a series of content and style images (the comparisons

[4] https://github.com/jcjohnson/neural-style
[5] https://github.com/anishathalye/neural-style
[6] https://github.com/alexjc/neural-doodle
[7] https://github.com/luanfujun/deep-photo-styletransfer

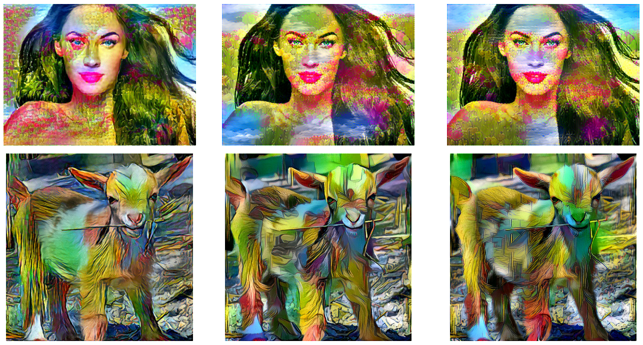
are not shown in this paper due to space limitations). Hence in the following, we take the L-BFGS implementation of Gatys-style as the main baseline against Lapstyle, and simply denote it as Gatys-style.

4.1 Qualititive Results

In Fig. 5, we qualitatively compare Lapstyle with Gatys-style and Photo-style on a variety of style and content images[8].

It can be observed that images synthesized by Lapstyle consistently improve on those by Gatys-style. In particular, more delicate details, e.g. contours and gradual color transitions are preserved, and much less artifacts are introduced from the style images. In the meantime, with the reinforced content details, the "stylishness" of these images do not degrade noticeably. This shows that the Laplacian constraint is a content constraint flexible enough to entertain style constraints.

The synthesized images by Photo-style have both advantages and disadvantages compared to Lapstyle. On the one hand, the synthesized images are smoother with less artifacts, for example the random patterns in the background now disappear. On the other hand, the synthesized images are less stylized and less natural, and the style transfer mostly happened to the colors. These results suggest that the Matting Laplacian is a strong content constraint which may expel the effects of the style constraints on local structures[9].



(a) 'conv4_2' (b) 'conv2_2' (c) 'conv2_2' + Lapstyle

Figure 6: Comparison of (a) Gatys-style with content layer = 'conv4_2', (b) Gatys-style with content layer = 'conv2_2', and (c) the Lapstyle extension of (b).

Lower-level content layer. One may argue that content details could be captured by using lower-level CNN filter banks as the content layer [6] than the default layer 'conv4_2', to achieve similar effects as does Lapstyle. Hence we experimented to use 'conv2_2' as the content layer. Fig. 6 compares the output images of using 'conv4_2', 'conv2_2' and 'conv2_2' + Lapstyle, respectively, given two test pairs appeared before. When 'conv2_2' is used as the content layer, the stylized images are apparently of lower quality than using 'conv4_2'. Nevertheless, incorporating the Laplacian loss still improves the stylized images in this setting. This verifies that the Laplacian filter, being a single manually chosen CNN filter, captures content details that can not be captured by the pretrained low-level CNN filter banks.

		Total loss	Laplacian loss	Content loss	Style loss
Init (Iter-0) losses		222.1 (185.2)	1.00 (0.00)	7.0 (3.3)	214.1 (184.1)
Lapstyle at Iter-1000	Losses	7.44 (5.58)	0.22 (0.33)	5.86 (4.44)	1.36 (0.93)
	Frac of total loss	/	2.96%	78.8%	18.3%
	Ratio to Init losses	3.3%	22%	84%	0.64%
Gatys-style at Iter-1000	Losses	11.41 (8.64)	4.49 (3.74)	5.73 (4.30)	1.19 (0.73)
	Frac of total loss	/	39.4%	50.2%	10.4%
	Ratio to Init losses	5.1%	449%	82%	0.56%
Ratios between Lapstyle and Gatys-style losses at Iter-1000		1.53	**0.049**	**1.02**	**1.14**

Table 1: The losses and relative changes with and without the Laplacian loss term in the optimization objective. The standard deviation of each loss is shown in parentheses. All the losses are normalized by the initial Laplacian loss.

Combining Multiple Laplacians. We experimented to combine two Laplacian losses with different pooling sizes. Compared with a single Laplacian loss, it led to improvement on some test images. Fig. 7 presents such an example, where the best stylization is achieved by combining a 4x4 pooled Laplacian loss with a 16x16 pooled Laplacian loss. The default setting of a single 4x4 pooled Laplacian loss failed to remove some artifacts on the girl's face.

However, the actual effects of combined Laplacians often depend on the particular content and style images. For fairness of comparison, we did not seek the best combinations for each test pair, instead in other experiments we simply used the default single 4x4 pooled Laplacian loss, which brings about empirically stable improvements.

Laplacian of Gaussian. The Laplacian of Gaussian filter (LoG, i.e., Gaussian smoothed Laplacian filter) [1] is commonly used in place of the vanilla Laplacian filter, as it is more robust to noisy pixel variations. We tested to replace the pooling + Laplacian structure with the commonly used LoG of size 5x5. As seen in Fig. 8, it led to much weaker effects of preserving detail structure and removing artifacts, compared to using an average pooling layer.

4.2 Quantitative Analysis

In order to quantitatively investigate how the Laplacian loss term impacts the Laplacian, content and style losses of the synthesized image, we computed the values of these losses with and without the Laplacian loss, respectively, averaged over the five pairs of input images in Fig. 5. The parameter settings were the same as described before. All the losses are normalized by the initial Laplacian loss. Table 1 lists the mean Laplacian, content, style and total losses in the first iteration and the last iteration of the optimization, respectively. It then compares the losses of the final images optimized with and without the Laplacian loss term.

[8] Some images are from [11, 17].

[9] Reducing the weight λ of the Matting Laplacian constrant does not substantially improve this problem.

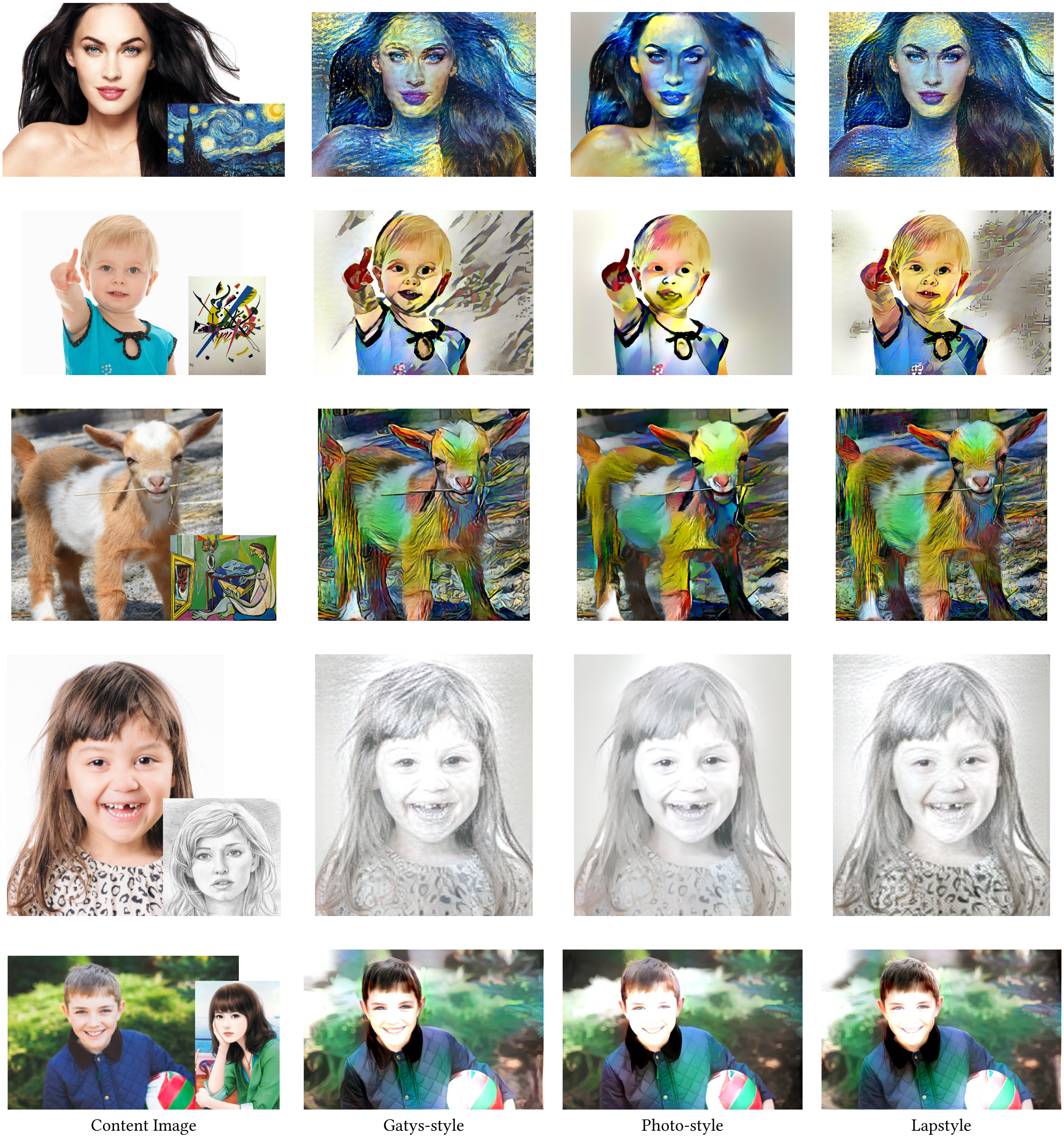


Figure 5: Comparisons of Gatys-style, Photo-style and Lapstyle. Compared to Gatys-style, Lapstyle always yields stylized images with finer and better-preserved local details, and less artifacts introduced from the style images. Photo-style yields images with least artifacts among the three methods, but the finer details are often lost.

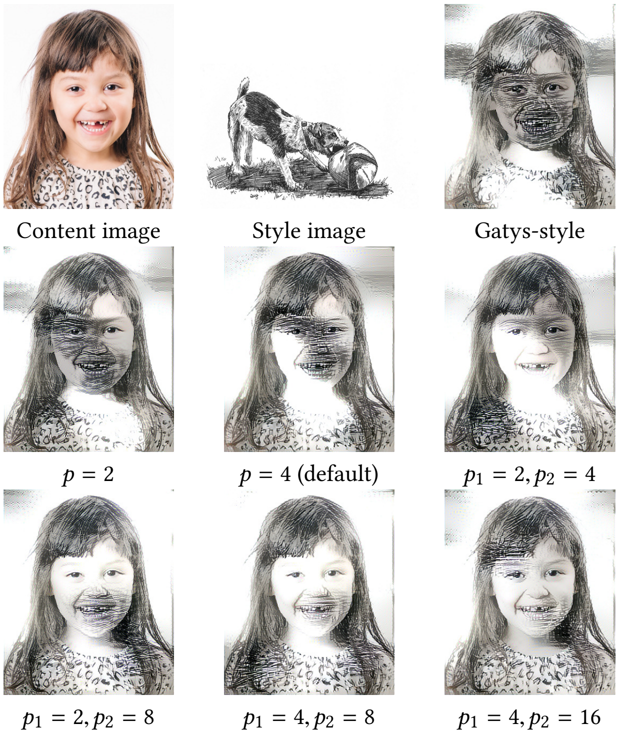


Figure 7: Results using one or two Laplacian losses of different pooling sizes. The captions of the 2nd and 3rd rows give the pooling layer sizes, e.g. $p_1 = 2, p_2 = 4$ means it uses a 2x2 pooled Laplacian loss combined with a 4x4 pooled Laplacian loss.

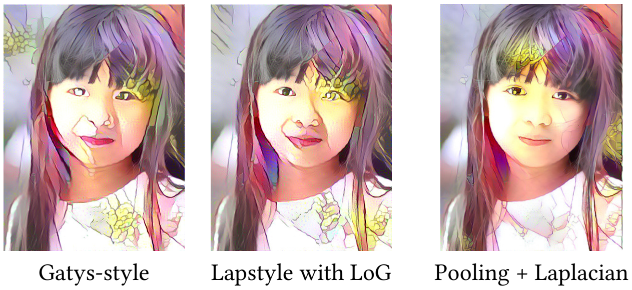


Figure 8: Comparison of the extensions with the Laplacian of Gaussian (LoG) filter and with pooling + Laplacian filter.

It can be seen from Table 1 that, with the Laplacian loss term, when the optimization finishes, the Laplacian loss was *reduced* to 22% of its initial value, amounting to only 2.96% of the total loss; as a trade-off, the content and style losses were 2% and 14% more than those of Gatys-style, respectively. Without this loss term, however, the final Laplacian loss *increased* to 449% of its initial value, amounting to 39.4% of the total loss, which may indirectly reflect the various artifacts in the stylized images.

The above quantitative analysis shows that by incorporating the Laplacian loss term, the Laplacian loss of the synthesized images is reduced to a large extent, at the cost of slight increases of the content and style losses. Thereby it can be concluded that 1) the image stylized by Lapstyle is almost equally "stylish" as that produced by the traditional Gatys-style method; 2) the content loss and the Laplacian loss govern independent aspects of the content of the stylized image.

5 CONCLUSIONS

Neural Style Transfer [6] adopts an optimization objective that is concerned to preserve only high-level CNN features of the content image. Due to the abstractiveness of these features, the synthesized images suffer from various artifacts and distortions. In this paper we have presented a method named Lapstyle that reduces artifacts and distortions, by incorporating a novel Laplacian loss term into the optimization objective. This loss encourages the stylized image to have a similar image Laplacian as that of the content image. The image Laplacian captures the detail structures (edges, contours, gradual color transitions, etc.) of the content image, and thus guides the optimization towards a synthesized image that more faithfully retains these details. We have empircally validated that images synthesized by Lapstyle look more natural and appealing, and contain much less artifacts than the traditional methods, while remaining almost equally "stylish". In addition, we have observed quantitatively that while the Laplacian loss is drastically reduced, the content and style losses only increase slightly. This confirms that the Laplacian loss term does not disrupt the image stylization.

In future work, we would like to explore ways to incorporate the Laplacian loss into Generative Adversarial Nets (GANs) [8, 28], to improve the quality of images generated by GANs.

ACKNOWLEGEMENT

This research was funded by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

REFERENCES

[1] Tinku Acharya and Ajoy K Ray. 2005. *Image processing: principles and applications.* John Wiley & Sons.

[2] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. 2004. Interactive digital photomontage. In *ACM Transactions on Graphics (ToG)*, Vol. 23. ACM, 294–302.

[3] A. J. Champandard. 2016. Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks. *ArXiv e-prints* (March 2016). arXiv:cs.CV/1603.01768

[4] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (ToG)* 31, 4 (2012), 82–1.

[5] Alexei A. Efros and Thomas K. Leung. 1999. Texture Synthesis by Non-Parametric Sampling. In *Proceedings of the International Conference on Computer Vision (ICCV) (ICCV '99)*. IEEE Computer Society, Washington, DC, USA, 1033–1038.

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2414–2423.

[7] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. 2017. Controlling Perceptual Factors in Neural Style Transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of Advances in neural information processing systems (NIPS)*. 2672–2680.

[9] Mark Hedley and Hong Yan. 1992. Segmentation of color images using spatial and color space information. *J. Electronic Imaging* 1, 4 (1992), 374–380.

[10] David J Heeger and James R Bergen. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 229–238.

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 694–711.

[12] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

[13] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. 2005. Texture Optimization for Example-based Synthesis. In *ACM SIGGRAPH (SIGGRAPH ’05)*. ACM, New York, NY, USA, 795–802.

[14] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (ToG)* 22, 3 (2003), 277–286.

[15] J. H. Lee, I. Choi, and M. H. Kim. 2016. Laplacian Patch-Based Image Synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2727–2735.

[16] Anat Levin, Dani Lischinski, and Yair Weiss. 2008. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 30, 2 (2008), 228–242.

[17] Chuan Li and Michael Wand. 2016. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2479–2486.

[18] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. 2017. Deep Photo Style Transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[19] Liqiang Nie, Meng Wang, Zhengjun Zha, Guangda Li, and Tat-Seng Chua. 2011. Multimedia Answering: Enriching Text QA with Media Information. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’11)*. ACM, 695–704.

[20] Liqiang Nie, Shuicheng Yan, Meng Wang, Richang Hong, and Tat-Seng Chua. 2012. Harvesting Visual Concepts for Image Search with Complex Queries. In *Proceedings of the 20th ACM International Conference on Multimedia (MM) (MM’12)*. ACM, 59–68.

[21] Y. Nikulin and R. Novak. 2016. Exploring the Neural Algorithm of Artistic Style. *ArXiv e-prints* (Feb. 2016). arXiv:cs.CV/1602.07188

[22] Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson Image Editing. In *ACM SIGGRAPH (SIGGRAPH ’03)*. ACM, New York, NY, USA, 313–318.

[23] Javier Portilla and Eero P Simoncelli. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision* 40, 1 (2000), 49–70.

[24] Simon JD Prince. 2012. *Computer vision: models, learning, and inference*. Cambridge University Press.

[25] Li-Yi Wei and Marc Levoy. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 479–488.

[26] Hanwang Zhang, Zawlin Kyaw, Shih-Fu Chang, and Tat-Seng Chua. 2017. Visual Translation Embedding Network for Visual Relation Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[27] Hanwang Zhang, Xindi Shang, Huanbo Luan, Meng Wang, and Tat-Seng Chua. 2016. Learning from collective intelligence: Feature learning using social images and tags. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 13 (2016).

[28] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.