Learning to Warp for Style Transfer

Figure 1. Our method performs non-parametric warping to match artistic geometric style. Output has comparable quality (or better) than state-of-the art, but warping is two orders of magnitude faster. The teaser shows content, style (geometry+texture), and output images for a Picasso style transfer (left) and a Salvaor Dali style transfer (right).

Xiao-Chang Liu Yong-Liang Yang Peter Hall
University of Bath
{XL2546,yy753,maspmh}@bath.ac.uk

Abstract

Since its inception in 2015, Style Transfer has focused on texturing a content image using an art exemplar. Recently, the geometric changes that artists make have been acknowledged as an important component of style [42, 55, 62, 63]. Our contribution is to propose a neural network that, uniquely, learns a mapping from a 4D array of inter-feature distances to a non-parametric 2D warp field. The system is generic in not being limited by semantic class, a single learned model will suffice; all examples in this paper are output from one model.

Our approach combines the benefits of the high speed of Liu et al. [42] with the non-parametric warping of Kim et al. [55]. Furthermore, our system extends the normal NST paradigm: although it can be used with a single exemplar, we also allow two style exemplars: one for texture and another for geometry. This supports far greater flexibility in use cases than single exemplars can provide.

1. Introduction

Neural style transfer (NST) is a current area of research in Non-photorealistic rendering (NPR), with applications in games, artistic design, architecture, and many other fields. By mimicking a wide gamut of artistic styles from visual examples, it greatly enriches the expressiveness of digital images. To reach its fullest extent, NST must be able to mimic not just the textural elements of style that are related to (*e.g.*, brush strokes), but geometric warps that artists use. This paper considers the problem of image stylization using deep neural networks, specifically focusing on artistic warping.

NST was first proposed by Gatys *et al.* [15], a paper set the paradigm for a great deal of work. The algorithm receives a *content image*, I_c, and an artistic *style exemplar*, I_s. These images provide the subject and rendering style for an output image: $I_o = \tau(I_c, I_s)$. The key idea is to construct a loss function of two parts, for content $\mathcal{L}_C(I_o, I_c)$ and for style $\mathcal{L}_S(I_o, I_s)$.

All of NST to date define both loss functions in terms of kernel responses, typically drawn from the convolutional layers within a network. The details of how the loss functions are formed and how the network is trained largely explain the development and diversification of current NST (see Section 2 for an overview). Such diversity notwithstanding, what is common among all the techniques is that the kernel responses depend on spatial color patterns and are spatially fixed. This means that NST can be regarded as a sophisticated form of tracing over the content image, which uses texture elicited from the style image to construct the artwork.

The approach is limiting because artists change the shape of the objects they render, that is: artists use geometric warping in their work. Warping is evident across all of art, here we give just a few examples among countless others. Caricaturists, such as Ronald Searle, exaggerate semantic features in an obvious way. English landscape artist George Stubbs painted bulls to look much larger, stronger, "beefier" than in real life – a kind of caricature. In his famous "Great Wave", Hokusai uses geometry to help emphasize natural power. Across the world, and in all times, cultures have employed geometrical distortions for expressive purpose. The art of children bears little correspondence with geometric reality, yet often remains

recognizable and is always cute. Figure 1 shows two artists for whom geometric changes are an intrinsic part of their style. Picasso took full advantage of the human capacity to recognize highly distorted objects as he helped invent 20th century Western art. Surrealist artist Salvador Dali's melting watches are instantly recognizable and attributable to him.

The importance of geometric warping is becoming recognized in the NST literature. Recent work has included geometric warping within an NST framework. The earliest of these are designed for single classes such as faces [63] or text [62]. Later work has provided more generic solutions [42, 55]. Our contribution is an NST architecture that performs a geometric warp and is uniquely characterized by the possession of *all* of the following properties:

- unlike Yaniv *et al*. [63] and WarpGan [52], it is not restricted to a single semantic class;
- unlike Kim *et al*. [55] who rely on forward and backward optimizations, we *train* a specifically designed feed-forward network to output warp fields given content and geometric images;
- warping is up-to two orders of magnitude faster than Kim *et al*. [55], while producing competitive results (see Section 4);
- unlike Liu *et al*. [42] who are limited to parametric warp fields, we produce a non-parametric warp;
- unlike every other NST algorithm other than Liu *et al*. [42], we support the use of two images to specify style, which adds versatility to image creation that is absent in other NST algorithms.

Our technique is explained in Section 3 but is easy to summarise: we warp an input image using our trained network, then apply regular NST.

To test the importance of geometric warping in human recognition of style, we performed an experiment. Details are provided in Section 4, but we summarise here. Given a style exemplar and outputs from two randomly selected NST algorithms, humans were asked to select the most similar pair, leaving the other as the odd-one-out. If neither NST algorithm used geometric warping, the style exemplar was the odd-one-out about 60% to 70% of the time, *i.e*., the two NST outputs were said to be more similar to one another than to the exemplar. In contrast, if one of the NST algorithms used geometric warping and the other did not, the non-warped image was odd-one-out between 60% and 70% of the time, leaving the non-warped NST as the odd-on-out. We emphasize that all of the outputs were subject to regular NST. This result shows that geometric warping is a major contributor to style recognition by humans, in addition to textural elements. The code is available at https://github.com/xch-liu/learning-warp-st.

2. Related Work

Prior to Neural Style Transfer (NST), Non-Photorealistic Rendering (NPR) algorithms were used to create artistic images. NPR algorithms accept 3D models, photographs, or videos as input, while no exemplar is necessary. NPR is capable of reaching many styles including Cubism [10], symbolic substitution [25], non-linear cameras that warp images [20], and caricature [2, 3]. The broader history of NPR is well documented elsewhere, see (*e.g*., [32]).

Most, but not all NPR algorithms are prescriptive, Image Analogies [23] being an exception. All NST algorithms learn, from Gatys *et al*. [15] onwards. The core innovation was to match the style of an output image to that of an exemplar. More specifically, the network receives a photographic *content image*, I_c, and an artistic *style exemplar*, I_s. It outputs the content in the style of the exemplar: $I_o = f(I_c, I_s)$.

2.1. Texture NST

Until very recently, Texture-NST has been the dominant, indeed sole, form of NST and was called NST with no further qualification. Jing *et al*. [27] use a full partition of (Texture) NST, which we follow.

Image-Optimization-Based Online: Methods in this category are characterized by transferring the style through iteratively optimizing an image. The first algorithm was proposed by Gatys *et al*. [15, 16]. They used the feature responses in higher layers of the VGG-Network [53] to represent the content of an image. The image style was represented by the feature correlations (also called Gram matrix) between different layers of the VGG. Some latter works used additional loss functions (*e.g*., Histogram loss [48] and Laplacian loss [34]) to help eliminate irregular artifacts. Li and Wand [33] were the first to propose an MRF-based NST algorithm.

Model-Optimization-Based Offline: Methods in this category optimize a generative model offline and generate the stylized image with a single forward pass at the testing stage. The first two algorithms were proposed by Johnson *et al*. [28] and Ulyanov *et al*. [58]. Ulyanov *et al*. [59] further replaced batch normalization with single image normalization and improved the stylization quality. However, the trained models in these methods are style-specific, which means separate models have to be trained for images with particular styles. To improve the flexibility, some works [6, 13, 35] incorporated multiple styles into one single model, or used one model to transfer arbitrary artistic style [18, 26, 36, 46, 56, 60, 61].

Variations of Texture NST: To date, NST has been extended for many different tasks (*e.g*., portrait painting style transfer [50], visual attribute transfer [9, 31, 39, 41, 64], semantic style transfer [4, 8, 45], video style transfer [5, 19, 24, 49], 3D style transfer [7, 29], and photorealistic style transfer [37, 43, 44]). Interested readers can refer to reviews [27, 51] and the explanation [38] on NST.

2.2. Geometric NST

There is a growing consensus that the geometric deformations artists used to make imaginative recreations of objects are worthy of consideration within NST. The literature is far

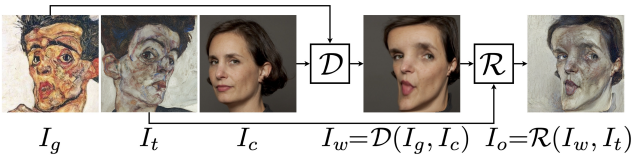

Figure 2. Our algorithm consists of two modules: $\mathcal{D}$ (geometric deformation) and $\mathcal{R}$ (texture rendering). For a given image pair$\{I_g, I_c\}$, module $\mathcal{D}$ estimates a pixel-wise warp field and computes the corresponding warping result I_w. Module $\mathcal{R}$ further renders I_w in the texture style of I_t. Artworks are by Egon Shiele.

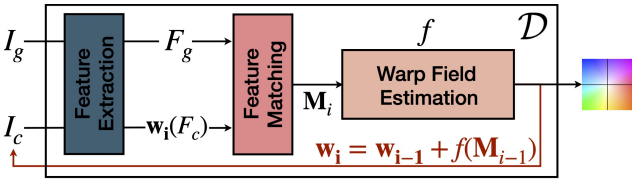

Figure 3. Geometric deformation module $\mathcal{D}$. Geometric exemplar and content image (I_g, I_c) are passed through the geometric deformation module to generate a pixel-wise warping field.

less voluminous than for texture NST, but the subject is no less important. Some methods are limited to specialized content domains such as faces [63], and text [62]. These methods produce excellent results. Yaniv [63] manages to characterize the geometric style of individual artists – analysis of style is absent from the literature otherwise, style is defined by example.

More recently Kim *et al*. [55] and Liu *et al*. [42] described more generic methods that operate over many classes. This added flexibility does not appear to cost much in terms of quality, see Section 4.

Kim *et al*. [55] proposed *Deformable Style Transfer*. They used Neural Best-Buddies (NBB) [1] to match points between the content image and the style exemplar, filtered matches with low activations, incorporated a warping loss in STROTSS-based texture style transfer [31]. While producing high-quality results, this method is computationally expensive since both NBB and STROTSS are optimization-based approaches and require back-and-forth passes through the pretrained network. Each step takes several minutes on a modern GPU.

Liu *et al*. [42] posited a mapping from a 4D function of distance measures, $M(i,j,k,l)$ to a 2D parametric warp field, $\mathbf{w}(i,j|\theta)$. Each $M(i,j,k,l)$ measures the distance between filter responses at two response locations, *i.e*., (i,j) in the content and (k,l) in the exemplar. The output is a 2D warp field covering the content image. The mapping is learned, making is very fast to use. Liu *et al*. [42] demonstrate their method with affine and bi-quadratic warps.

The approach we present in this paper has the speed of Liu *et al*. [42], yet supports the same arbitrary deformations as DST [55]. DST [55] is akin to Image Optimization, we use Model Optimization.

3. Geometric & Texture Style Transfer

The inputs to our neural style transfer algorithm are: 1) a content image I_c to be transferred, 2) an exemplar I_g to guide geometric transfer, and 3) an exemplar I_t to guide texture transfer. Note that we can set $I_g = I_t$, so that a single style exemplar is sufficient.

As shown in Figure 2, our neural style transfer algorithm contains two main modules. Our *geometric warping* module $\mathcal{D}$ computes a non-parametric vector field to warp the content image I_c to match the geometric style in the exemplar I_g. The *texture rendering* module $\mathcal{R}$ uses the texture exemplar I_t to produce the final result I_o. The modules are independent, but in use need to be ordered. Rendering texture which is then warping also warps the texture, we prefer to warp the content before texturing. All the outputs in this paper we produced under the following model:

$$I_o = \mathcal{R}(\mathcal{D}(I_g, I_c), I_t). \tag{1}$$

The warp module $\mathcal{D}$ and the texture module $\mathcal{R}$ are detailed in Section 3.1 and Section 3.2, respectively.

3.1. Geometric Style

The role of module $\mathcal{D}$ is to warp the content image I_c to match the geometric exemplar I_g. The key idea is to train a neural network that is able to infer a two-dimensional warp field $\mathbf{w}$ given a four-dimensional scalar function $\mathbf{M}$ that measures feature similarity. As shown in Figure 3, the module has three major components: 1) *feature extraction* to get features F_c from I_c, and F_g from I_g; 2) *feature correlation* to measure feature similarity $\mathbf{M}(F_c, F_g)$; and 3) training a *warp network* to output a function f such that $\mathbf{w} = f(\mathbf{M})$. Once trained, the network f can be used on new inputs without modification. All outputs in this paper were produced with a single warp network.

Note that the warp field is determined by $\mathbf{w}$ is non-parametric. Also, training and using the network are very efficient (see Section 4). The approach is not limited to a narrow range of semantic content, (*e.g*., faces, text), yet combines the advantages of the diverse deformations of Kim *et al*. [55] and the computational efficiency of Liu *et al*. [42]. In the following, we elaborate on each of the three components in detail.

3.1.1 Feature Extraction

Like many NST algorithms, we use the VGG network [53], trained for object recognition as a feature source. We extract features from *pool*4 layer of VGG, followed by an L2-normalization. The output is a feature field F of size $W \times H$. This is 16×16 in our case, which balances computational efficiency with warp quality. Each $F(i,j)$ is an N dimensional vector of unit length. We use this network on the content image I_c and the geometric style exemplar I_g to get feature fields F_c and F_g, respectively.

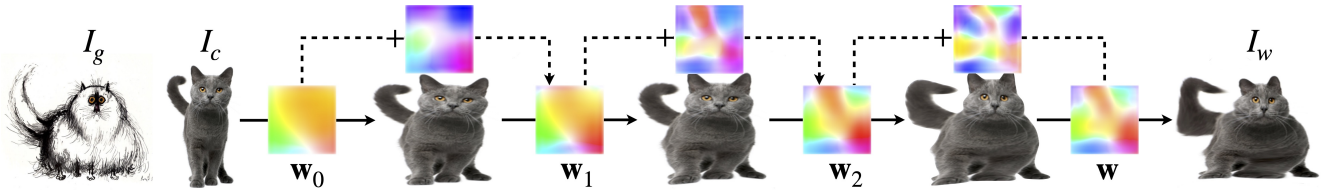

Figure 4. Warp field estimation. The warp field is iteratively estimated by repeating the forward propagation. The upper row shows the predicted field after each forward propagation, the final $\mathbf{w}$ was got by adding them to the initial $\mathbf{w}_0$. The lower row shows the intermediate results. Cat target is by Ronald Searle.

3.1.2 Feature Correlation

This component computes feature correlation scores between every position (i,j) in F_c and every position (k,l) in F_g. The result is stored in a four dimensional scalar function $\mathbf{M} \in \mathbb{R}^{W\times H\times W\times H}$. Each element $\mathbf{M}_{c,g}(i,j,k,l)$ is computed as:

$$\mathbf{M}_{cg}(i,j,k,l) = \frac{\langle F_c(i,j)|F_g(k,l)\rangle}{\sqrt{\sum_{p=1}^{W}\sum_{q=1}^{H}\langle F_c(i,j)|F_g(p,q)\rangle^2}}, \quad (2)$$

where $\langle F_c(i,j)|F_g(k,l)\rangle$ is the inner product between vectors. This form of correlation volume has been used in tasks such as flow estimation [12, 57], correspondence estimation [30, 66], and exemplar-based colorization [65].

3.1.3 Warp Network: Training and Using.

Our key technical contribution is to train a neural network f to output a non-parametric warp field $\mathbf{w}$, given a four dimensional correlation volume $\mathbf{M}$. We can write this as $\mathbf{w} = f(\mathbf{M})$. Formally, the network f is a mapping as follows:

$$f : \Re^{W\times H\times W\times H} \mapsto \Re^{W_I\times H_I\times 2}, \quad (3)$$

with W, H being the size of the feature arrays, and W_I, H_I the image size. The warp field is used to warp the content image to get $\mathbf{w}[I_c]$, which is the output of the warp module $\mathcal{D}$.

In principle, training is not required as it is sufficient to solve an optimization problem of the form $\min_{\mathbf{w}} h(\mathbf{w}[I_c], I_g)$ for any pair of images I_c and I_g, with h a measuring function. This optimization-based approach is followed by Kim *et al.* [55], and we optimize during training. But per-instance optimization is slow when compared to computing the warp field directly from a trained network f. Our results show that direct computation is two orders of magnitude faster than Kim *et al.* [55] (see Table 2 in Section 4).

Training: Our network is trained with a set of image pairs that are semantically related or have geometrically similar parts. The image pairs cover a wide range of semantic content: faces, animals, and so on. To improve the model's generalization on artistic domains, we use artistic augmentation to create a texture-augmented copy of every training image. Once trained the deformation network can be applied to any image regardless of its semantic content.

The underlying idea is to locally move pixels in the content image and (re-)compute features in the newly warped image until a loss function is minimized. More specifically, let F^m denote the elements within a feature field that are influenced by pixel m. Let $\mathbf{w}(F_c)$ denote the content feature field after the content image is warped. Let $\mathbf{M}(\mathbf{w}(F_c), F_g)$ denote the measure field computed after the warp. The loss function is specified to be:

$$\mathcal{L}(F_c, F_g|\mathbf{w}) = -\sum_{m\in I_c}\sum_{n\in\mathcal{N}_m} \log(p(\mathbf{w}(F_c^m), F_g^n)), \quad (4)$$

where $\mathcal{N}_m$ is a search window centered on m, we use a 9×9 region. $p(.,.)$ is the probability that two features should be classified together, we use the softmax function:

$$p(\mathbf{w}(F_c^m), F_g^n) = \frac{\exp(\mathbf{M}(\mathbf{w}(F_c^m), F_g^n)}{\sum_{t\in\mathcal{N}_m}\exp(\mathbf{M}(\mathbf{w}(F_c^m), F_g^t)}. \quad (5)$$

The goal of training is to find the network parameters (*i.e.*, its connection weights) that minimise the loss. The derivatives of $\mathcal{L}$ with respect to the warp field $\mathbf{w}$ ($\partial\mathcal{L}/\partial\mathbf{w}(F_c)$, $\partial\mathbf{w}(F_c)/\partial\mathbf{w}$) can be back-propagated into the warp net f to learn parameters.

Improving the Result: To achieve a high-precision estimation, we iteratively refine the warp field during training. At each step we use the current warp field $\mathbf{w}_i$ to transform the content image, then (re-)extract the features F_c so that a new measure $\mathbf{M}$ can be computed. Notice that each step estimates the change from the previous step, which is a differential. We express this as:

$$\mathbf{w}_i - \mathbf{w}_{i-1} = f(\mathbf{M}(\mathbf{w}_{i-1}(F_c), F_g)), \quad (6)$$

where $\mathbf{w}_i$ represents the estimated transformation field at the i^{th} iteration. The final transformation field $\mathbf{w}$ is the accumulated differential fields:

$$\mathbf{w} = \mathbf{w}_0 + \sum_{k=0}^{K-1} f(\mathbf{M}(\mathbf{w}_k(F_c), F_g)), \quad (7)$$

where $\mathbf{w}_0$ is the initial transformation field which was computed through Equation 3 using the original feature pair$\{F_c, F_g\}$, and K is the chosen number of iterations (in practice we find $K = 3$ is enough, giving four fields $\mathbf{w}_0$ to $\mathbf{w}_3$). As shown in Figure 4, the estimated transformation fields become increasingly accurate with respect to the geometric exemplar.

Network-based Warping: Once trained, the network will directly compute a warp field given a pair of feature maps. Each pass takes about 0.3 seconds. Multiple passes can be used to increase accuracy, as described above. We stop iterating when the result changes little, or after 4 passes, whichever is sooner. Four passes consume 1.2 seconds, compared with about 6 seconds for a single pair during training, and between 80 and 133 seconds for using Kim *et al.*'s optimisation [55]. Table 2 has more details.

3.2. Texture Style

In this sub-section, we detail the texture rendering stage with module $\mathcal{R}$. As shown in Figure 2, this module accepts warped image I_w and texture exemplar I_t as input, to yield an output image: $I_o = \mathcal{R}(I_w, I_t)$.

In line with the majority of the NST literature, we formulate this as an optimization task to minimize both content loss $\Delta_C(I_o, I_w)$, and texture style loss $\Delta_S(I_o, I_t)$, both of which depend on feature maps from a neural network trained for object detection. Our only change is to adopt a coarse-to-fine strategy that preferentially transfers texture with increasing details into different areas of the output image. This strategy has been used for decades in prescriptive texture synthesis [22, 47], and more recently in texture-only NST [14, 17, 54] where it helps improve the style transfer results. In our work, we leverage it to resolve blur and other artifacts that would otherwise occur due to geometric warping.

We follow the parametric modeling strategy proposed by Gatys *et al.* [16] to represent texture style and content in the domain of a CNN. Specifically, we use a Gram-based representation, which is the correlation between filter responses in different layers of VGG, to model textures. The content representation is relatively straightforward, we use the inter-layer filter response directly.

Denote the feature activation map of input image I at layer l of VGG by $\mathcal{F}^l(I)$. This map is of size $W_l \times H_l$, and each feature element is a vector of C_l components corresponding to the number of channels. Then the texture style of image I at layer l can be represented by the Gram matrix as:

$$\mathbf{G}(\mathcal{F}^l(I)) = [\mathcal{F}^l(I)]^\intercal [\mathcal{F}^l(I)], \tag{8}$$

where $[\mathcal{F}^l(I)]$ is the reformatted feature map such that each feature is a row vector in a matrix of $W_l \times H_l$ columns, and G is a $C_l \times C_l$ symmetric matrix. The texture style distance is specified to be:

$$\Delta_S(I_o, I_t) = \sum_{l \in l_t} \left\| \mathbf{G}(\mathcal{F}^l(I_o)) - \mathbf{G}(\mathcal{F}^l(I_t)) \right\|^2, \tag{9}$$

where l_t is the set of selected layers for texture style representation. The content distance is specified to be the L2-norm between feature maps:

$$\Delta_C(I_o, I_w) = \left\| \mathcal{F}^{l_c}(I_o) - \mathcal{F}^{l_c}(I_w) \right\|^2, \tag{10}$$

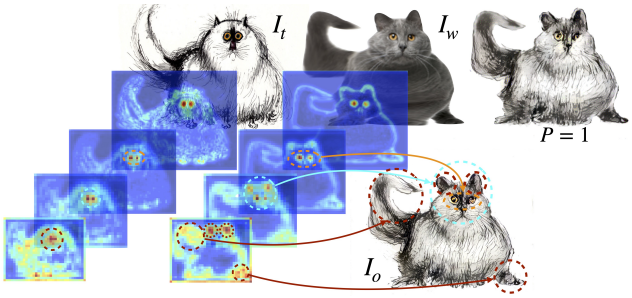

Figure 5. Texture rendering at multiple scales: larger scales fill regions stretched by warping, smaller scales deal with compressed regions with fruitful details. The complement of multiple scales improves the rendering quality. Upper right shows the image rendered with a single pyramid layer, $P = 1$.

where l_c is the selected layer for content representation.

Texture style transfer is then instantiated as the following optimization problem:

$$I_o = \underset{I}{\operatorname{argmin}} \left[\alpha \Delta_S(I, I_t) + \beta \Delta_C(I, I_w) \right], \tag{11}$$

where α and β are the balancing weights used to control the extent of stylized effects.

Multi-scale Strategy. We apply the texture rendering process to the images involved (I_o, I_w, and I_t) at multiple scales. Images at different scales are obtained by feeding them into a Gaussian pyramid, where each pyramid layer is formed by blurring and downsampling the previous layer. Let I^p, I^p_w, and I^p_t be the images at the p^{th} scale of the Gaussian pyramid. Rather than solve Equation 11 for each layer, we solve across all scales:

$$I_o = \underset{I}{\operatorname{argmin}} \sum_{p=0}^{P-1} \alpha \Delta_S(I^p, I^p_w) + \beta \Delta_C(I^p, I^p_t), \tag{12}$$

where P is the number of scales (we use $P = 4$).

As shown in Figure 5, higher pyramids level will compensate and strengthen regions that are not well covered by lower levels (typically where an image region has been stretched). On the other hand, low-levels fill in the small-scale details that the higher levels tend to blur (where regions have been compressed).

3.3. Implementation

The geometric warping network is trained with images from PF-PASCAL [21] and MS COCO [40]. All images are resized to 256×256. We trained the network with batch size 16 and learning rate 1×10^{-5}. Training takes about two hours on a single GPU. Please see the supplementary for a detailed description of the architecture of the warp network f. After warping, empty background regions are inpainted. For the texture rendering module, we compute content distance at layer $relu4\_2$ and texture distance at layers $relu1\_1$, $relu2\_1$, $relu3\_1$, $relu4\_1$, and $relu5\_1$.

Content Style NST [16] AdaIN [26] DST [55] Ours

Figure 6. By using a single style exemplar, we compare style transfer results with Gatys *et al*. [16], AdaIN [26], and DST [55]. Texture-based NST methods [16, 26] do not change the geometric style of the content image. In contrast, both DST [55] and our approach deform the content image to match the geometry of the style exemplar. Our method not only warps, but positions and proportions the main object to be consistent with the style exemplar. From top to bottom the artists are: Picasso, Bacon, an anonymous child, and Hokusai.

4. Results

We present qualitative and quantitative results, over a variety of artistic styles and comparing with several alternative methods. All results are generated from the same trained deformation module. Further results, including tests of our method's performance on annotated dataset and robustness to the artistic domain, can be found in the supplementary material.

4.1. Qualitative Comparisons

We provide images for qualitative comparison against both generic and class-specific NST algorithms. The class-specific geometric NST algorithms rely on extensively trained models – there is no guarantee a model we train would reproduce the results. Therefore we have used their results directly from their literature, producing our results using the same source material.

General Comparison. Figure 6 allows readers to gauge the impact of geometric warping. It shows texture-only NST [16, 26] alongside the non-parametric warped outputs from both DST [55] and our system. We believe is easy to see the effectiveness of geometry transfer. The texture-only NST methods [16, 26] fail to capture the shape changes that are an inherent part of the style. This limits their capacity for mimicry to artistic styles that exhibit no geometric warping. In contrast, DST [55] and our method are capable of capturing the geometric style of the exemplar images. Our coarse-to-fine warping strategy achieves better deformation on multiple object classes, such as the head shape/posture of portraits and the size/proportion of the main objects. Meanwhile, the texture styles are also well transferred.

Non-parametric vs. Parametric Warp. GST [42] provides a neural architecture for geometric warping, but is limited to global bilinear warps. Figure 7 compares the output from GST with ours, using a cow image as content and a bull painting by Stubbs as the single exemplar. GST increases the body mass

Content Style Ours GST

Figure 7. Deformation comparison with GST [42]. The results are directly adopted from their paper. An animal painting by Stubbs is used as a single exemplar. Due to non-parametric warp, our method better transfers the head in terms of its size compared with the body.

Content Style DST [55] FoA [63] Ours

Figure 8. Comparison of our method with DST and FoA [63] on face art. The style images are from Leger.

Content Style DST WarpGAN Ours

Figure 9. Comparison of our method with DST [55] and WarpGAN [52] on facial caricature.

of the cow, which is in line with the geometric style of Stubbs, who painted bulls to look bigger and stronger than the actual case. However, because their parametric warp is global, the cow's head also increases in size – but should not. Our result is closer to Stubbs's style in keeping the head size small compared to the body.

Face-of-Art and WarpGAN. The ST literature includes geometric warping designed for specific cases. Yaniv *et al*. [63] use a strong model of faces (a point distribution model [11]) that needs to be trained with many examples from the same artist. The model can be analyzed to elicit artistic style, and high-quality images can be generated. However, the use of a strong model restricts its scope to a single semantic class. Figure 8 shows the comparison with DST [55] and FoA [63] using portraits by Fernand Leger as the style exemplar. All methods echo the geometric style of the exemplar, although in slightly different ways; texture transfer differs too. The final output is of high quality in all cases.

WarpGAN [52] is designed specifically for stylized portraits or caricatures. It falls into the category of *collection style transfer*, in which the target style is defined by a collection of images rather than one. In contrast, our approach and DST belong to *example-guided style transfer*, in which the target style comes from a single example. This means only our approach and DST can handle every single content/style image pair. Figure 9 compares our results, outputs of DST taken from [55] and outputs of WarpGAN taken from [52]. All methods produce high-quality results. Our coarse-to-fine warp strategy helps to preserve details such as face contours, eyebrow shapes, *etc*.

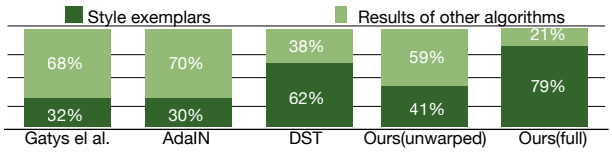

Table 1. Subjective similarity.

4.2. Quantitative Comparisons

Here we provide comparison results from quantitative experiments, to assess the subjective similarity between the output and GPU running times.

Subjective Similarity. The aim of NST is to transfer style, here we ask "how well the style was transferred?" To quantitatively gauge this subjective assessment, we performed an on-line similarity experiment. Each of the 50 participants was shown a style exemplar and two output images from NST algorithms as listed below. The three images were presented on a row in random order. The participant was asked to pick two they judged to be the most similar. Each participant repeated this 5 times.

The output images were created using five different NST methods: Gatys *et al*. [16], AdaIN [26], DST [55], our method without warping, and our method with warping. We used ten content-style pairs to generate results. In each case a single exemplar was used to represent both geometry and texture styles. In this way, we created a group of five outputs for each exemplar. At each trial, the exemplar was selected at random, and then two output images from its group were selected at random. We obtained 25 votes for each pair of methods.

This experiment is similar to those conducted by Liu *et al*. [42] and by Kim *et al*. [55], although the latter also asked about the preservation of content. We have yet to find an example where the warp is so extreme as to make content unrecognizable, so we opted for the simpler, one question experiment. As shown in Table 1, our results agree with the previous literature: geometric warping has a significant impact on subjective similarity. Our method achieved the highest user

Methods	**Runtime (s)**			
	Geometric Warping	Texture Rendering		
		256^2	512^2	1024^2
Gatys *et al*. [16]	N/A	14	33	116
AdaIN [26]	N/A	0.037	0.14	0.55
DST [55]	83–133	62	100	165
Ours	0.3–1.2	16	48	140

Table 2. Computational efficiency comparison on RTX 2080 Ti. Running times are in seconds. Artistic warping is not applicable to [16, 26].

preference, *i.e.*, is subjectively deemed closer to the target style.

GPU Time. In Table 2 we compare the running time of our method and [16, 26, 55] for several image sizes. Note that the geometric warpings of DST [55] and our method are all independent of the image size. The warping of DST consists of two steps: (1) find matching points with Neural Best-Buddies [1], which takes about two minutes on a GPU; and (2) clean (NBB) points, which takes a few seconds. For texture rendering, compared to model optimization based offline method [26], [16, 55] and our work are advantageous for quality at the cost of speed. Our geometric warping module could be combined with offline NST methods such as [26].

5. Discussion

In this section, we discuss potential applications of our method, followed by its limitations.

Applications. Since we explicitly model both geometry and texture styles, we can easily use two exemplars to produce an output as $I_o = \mu(I_c, I_g, I_t)$. This provides the potential for previously unavailable versatility and control. Figure 10 shows several practical examples benefiting from separated geometry and texture style transfer.

The first example uses a face and an African mask made of clay, along with two different textures to produce two different outputs. One warps the face to the mask, which is then textured using a marble example to create an image of a statuette. The second example warps the mask onto the face, which is then textured with wood to create a wooden mask to snugly fit the face. This shows that the model learned by the network "goes both ways". Our second example is inspired by Picasso's famous remark, "Every child is an artist. The problem is how to remain an artist once he grows up." Child art is notoriously difficult – perhaps impossible – for adult artists to reproduce. Our approach makes it possible to emulate child art using child art exemplars. In this case, we have warped a chicken to a child's drawing, which we then "crayoned" over. A third example is virtual try-on. In our example, a dress that might be bought is warped onto the dress being worn, and then textured. This is not an art example, rather it shows that applications of our system may extend beyond its original design intent.

Figure 10. Example applications. Left: a face, clay mask, and two textures make a wooden mask and marble statuette. Top: emulating a child's crayon drawing. Bottom: virtual try-on.

Figure 11. The in-principle limit is the 1-1 mapping assumption. Left: A low-feature count in the Matisse (detail) leads to unexpected results in the warped dancer. Middle: Hindu god Brahma, with many similar faces having too many features. Right: a cycle of starfish, doughnut, eight-shaped octopus cause failures due to topological differences.

Limitations. Our approach is limited by its assumptions in terms of both geometric and texture transfer. The limitations on texture transfer are shared with many other NST algorithms. We will focus on discussing geometric transfer, since our contribution is in geometric warping.

The key limiting assumption is that *the content image and geometric exemplar each exhibit local discriminative features that can be mapped 1-1*. The mapping struggles when the geometric exemplar has too few features, or has too many nearly identical features. Both of these cases are shown in Figure 11. Another interesting failure case occurs when the topology of the shapes involved differ, again shown. All of these are fundamental in that they will require changes to our algorithm to address.

The reader may be surprised we have not included cases where the semantic content of source and target differ. Output can vary in such case, but acceptability is a value judgment that depends on the intentions of the user. We anticipate that most people wish to follow the majority of artistic practice and deform objects within semantic class limits, most of the time. They will rarely if ever wish to warp an owl into a house, for example, and if they choose to do so, the output may be acceptable (there may be some artistic reason to have a house-shaped owl). This discussion is expanded upon in the supplementary, with examples.

6. Conclusion

Our paper presents a novel method for neutral style transfer with more flexible and efficient non-parametric geometric deformations. While generating competitive results, our method significantly improves speed. The impact of geometric warping on style is clear – warping is needed to better mimic many artistic styles. Partitioning texture from geometry allows greater flexibility in use allowing two exemplars to influence the outcome, including potential applications beyond NST.

Acknowledgement. This work was partially funded by the China Scholarship Council under Grant No. 201906200059.

References

[1] Kfir Aberman, Jing Liao, Mingyi Shi, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Neural best-buddies: Sparse cross-domain correspondence. *ACM Transactions on Graphics (TOG)*, 37(4):69, 2018.

[2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1999.

[3] Susan E Brennan. Caricature generator: The dynamic exaggeration of faces by computer. *Leonardo*, 18(3):170–178, 1985.

[4] Alex J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. `arXiv:1603.01768 [cs.CV]`, 2016.

[5] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[6] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. StyleBank: an explicit representation for neural image style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stereoscopic neural style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] Yi-Lei Chen and Chiou-Ting Hsu. Towards deep style transfer: A content-aware perspective. In *British Machine Vision Conference (BMVC)*, 2016.

[9] Ming-Ming Cheng, Xiao-Chang Liu, Jie Wang, Shao-Ping Lu, Yu-Kun Lai, and Paul L. Rosin. Structure-preserving neural style transfer. *IEEE Transactions on Image Processing*, 29:909–920, 2020.

[10] John P Collomosse and Peter M Hall. Cubist style rendering from photographs. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):443–453, 2003.

[11] Timothy F Cootes and Christopher J Taylor. Active shape models—'smart snakes'. In *British Machine Vision Conference (BMVC)*, pages 266–275, 1992.

[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[13] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *The International Conference on Learning Representations (ICLR)*, 2017.

[14] Michael Elad and Peyman Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26(5):2338–2351, 2017.

[15] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. `arXiv:1508.06576[cs.CV]`, 2015.

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

[17] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[18] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *British Machine Vision Conference (BMVC)*, 2017.

[19] A. Gupta, Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Characterizing and improving stability in neural style transfer. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[20] Peter M Hall, John P Collomosse, Yi-Zhe Song, Peiyi Shen, and Chuan Li. Rtcams: A new perspective on nonphotorealistic rendering from photographs. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):966–979, 2007.

[21] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow: Semantic correspondences from object proposals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(7):1711–1725, 2017.

[22] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1995.

[23] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2001.

[24] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. Real-time neural style transfer for videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] Hua Huang, Lei Zhang, and Hong-Chao Zhang. Arcimboldo-like collage using internet images. *ACM Transactions on Graphics (TOG)*, 30(6):155, 2011.

[26] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[27] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385, 2020.

[28] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016.

[29] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] Seungryong Kim, Stephen Lin, Sang Ryul Jeon, Dongbo Min, and Kwanghoon Sohn. Recurrent transformer networks for semantic correspondence. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[31] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[32] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885, 2013.

[33] Chuan Li and Michael Wand. Combining Markov random fields and convolutional neural networks for image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. Laplacian-steered neural style transfer. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, 2017.

[35] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[37] Y. Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *European Conference on Computer Vision (ECCV)*, 2018.

[38] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[39] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Transactions on Graphics (TOG)*, 36(4):120:1–120:15, 2017.

[40] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[41] Xiao-Chang Liu, Ming-Ming Cheng, Yu-Kun Lai, and Paul L. Rosin. Depth-aware neural style transfer. In *Non-Photorealistic Animation and Rendering (NPAR)*, 2017.

[42] Xiao-Chang Liu, Xuan-Yi Li, Ming-Ming Cheng, and Peter Hall. Geometric style transfer. `arXiv:2007.0547[cs.CV]`, 2020.

[43] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[44] Roey Mechrez, Eli Shechtman, and Lihi Zelnik-Manor. Photo-realistic style transfer with screened poisson equation. In *British Machine Vision Conference (BMVC)*, 2017.

[45] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *European Conference on Computer Vision (ECCV)*, 2018.

[46] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[48] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. `arXiv:1701.08893 [cs.GR]`, 2017.

[49] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, 126:1199–1219, 2018.

[50] Ahmed Selim, Mohamed Elgharib, and Linda Doyle. Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 35(4):129, 2016.

[51] Amir Semmo, Tobias Isenberg, and Jürgen Döllner. Neural style transfer: a paradigm shift for image-based artistic rendering? In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 2017.

[52] Yichun Shi, Debayan Deb, and Anil K Jain. Warpgan: Automatic caricature generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *The International Conference on Learning Representations (ICLR)*, 2015.

[54] Xavier Snelgrove. High-resolution multi-scale neural texture synthesis. In *SIGGRAPH Asia 2017 Technical Briefs*, 2017.

[55] S. Y. Kim Sunnie, Kolkin Nicholas, Salavon Jason, and Shakhnarovich Gregory. Deformable style transfer. In *European Conference on Computer Vision (ECCV)*, 2020.

[56] Jan Svoboda, Asha Anoosheh, Christian Osendorfer, and Jonathan Masci. Two-stage peer-regularized feature recombination for arbitrary image style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[57] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, 2020.

[58] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *The International Conference on Machine Learning(ICML)*, 2016.

[59] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[60] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative distillation for ultra-resolution universal style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[61] Zhizhong Wang, Lei Zhao, Haibo Chen, Lihong Qiu, Qihang Mo, Sihuan Lin, Wei Xing, and Dongming Lu. Diversified arbitrary style transfer via deep feature perturbation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[62] Shuai Yang, Zhangyang Wang, Zhaowen Wang, Ning Xu, Jiaying Liu, and Zongming Guo. Controllable artistic text style transfer via shape-matching gan. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[63] Jordan Yaniv, Yael Newman, and Ariel Shamir. The face of art: Landmark detection and geometric style in portraits. *ACM Transactions on Graphics (TOG)*, 38(4):60, 2019.

[64] Yuan Yao, Jianqiang Ren, Xuansong Xie, Weidong Liu, Yong-Jin Liu, and Jun Wang. Attention-aware multi-stroke style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[65] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[66] Pan Zhang, Bo Zhang, Dong Chen, Lu Yuan, and Fang Wen. Cross-domain correspondence learning for exemplar-based image translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.