Eurographics Symposium on Rendering 2018
T. Hachisuka and W. Jakob
(Guest Editors)

Volume 37 (2018), Number 4

Deep Painterly Harmonization

Fujun Luan[1] Sylvain Paris[2] Eli Shechtman[2] Kavita Bala[1]

[1]Cornell University [2]Adobe Research

Figure 1: *Our method automatically harmonizes the compositing of an element into a painting. Given the proposed painting and element on the left, we show the compositing results (cropped for best fit) of unadjusted cut-and-paste, Deep Image Analogy [LYY*17], and our method.*

Abstract

Copying an element from a photo and pasting it into a painting is a challenging task. Applying photo compositing techniques in this context yields subpar results that look like a collage — and existing painterly stylization algorithms, which are global, perform poorly when applied locally. We address these issues with a dedicated algorithm that carefully determines the local statistics to be transferred. We ensure both spatial and inter-scale statistical consistency and demonstrate that both aspects are key to generating quality results. To cope with the diversity of abstraction levels and types of paintings, we introduce a technique to adjust the parameters of the transfer depending on the painting. We show that our algorithm produces significantly better results than photo compositing or global stylization techniques and that it enables creative painterly edits that would be otherwise difficult to achieve.

CCS Concepts

•***Computing methodologies*** → ***Image processing;***

1. Introduction

Image compositing is a key operation to create new visual content. It allows artists to remix existing materials into new pieces and artists such as Man Ray and David Hockney have created masterpieces using this technique. Compositing can be used in different contexts. In applications like photo collage, visible seams are desirable. But in others, the objective is to make the compositing inconspicuous, for instance, to add an object into a photograph in a way that makes it look like the object was present in the original scene. Many tools have been developed for photographic compositing, e.g., to remove boundary seams [PGB03], match the color [XADR12] or also fine texture [SJMP10]. However, there is no equivalent for paintings. If one seeks to add an object into a painting, the options are limited. One can paint the object manually or with a painting engine [CKIW15] but this requires time and skills that few people have. As we shall see, resorting to algorithms designed for photographs produces subpar results because they do not handle the brush texture and abstraction typical of paintings. And applying existing painterly stylization algorithms as is also performs poorly because they are meant for global styliza-

tion whereas we seek a local harmonization of color, texture, and structure properties.

In this paper, we address these challenges and enable one to copy an object in a photo and paste it into a painting so that the composite still looks like a genuine painting in the style of the original painting. We build upon recent work on painterly stylization [GEB16] to harmonize the appearance of the pasted object so that it matches that of the painting. Our strategy is to transfer *relevant* statistics of neural responses from the painting to the pasted object, with the main contribution being how we determine which statistics to transfer. Akin to previous work, we use the responses of the VGG neural network [SZ14] for the statistics that drive the process. In this context, we show that spatial consistency and inter-scale consistency matter. That is, transferring statistics that come from a small set of regions in the painting yields better results than using many isolated locations. Further, preserving the correlation of the neural responses between the layers of the network also improves the output quality. To achieve these two objectives, we introduce a two-pass algorithm: the first pass achieves coarse harmonization at a single scale. This serves as a starting point for the second pass which implements a fine multi-scale refinement. Figure 1(right) shows the results from our approach compared to a related technique.

We demonstrate our approach on a variety of examples. Painterly compositing is a demanding task because the synthesized style is juxtaposed with the original painting, making any discrepancy immediately visible. As a consequence, results from global stylization techniques that may be satisfying when observed in isolation can be disappointing in the context of compositing because the inherent side-by-side comparison with the original painting makes it easy to identify even subtle differences. In contrast, we conducted a user study that shows that our algorithm produces composites that are often perceived as genuine paintings.

1.1. Related Work

Image Harmonization. The simplest way to blend images is to combine the foreground and background color values using linear interpolation, which is often accomplished using alpha matting [PD84]. Gradient-domain compositing (or Poisson blending) was first introduced by Pérez et al. [PGB03] which considers the boundary condition for seamless cloning. Xue et al. [XADR12] identified key statistical factors that affect the realism of photo compositings such as luminance, color temperature, saturation, and local contrast, and matched the histograms accordingly. Deep neural networks [ZKSE15, TSL*17] further improved color properties of the composite by learning to improve the overall photo realism. Multi-Scale Image Harmonization [SJMP10] introduced smooth histogram and noise matching which handles fine texture on top of color, however it does not capture more structured textures like brush strokes which often appear in paintings. Image Melding [DSB*12] combines Poisson blending with patch-based synthesis [BSFG09] in a unified optimization framework to harmonize color and patch similarity. Camouflage Images [CHM*10] proposed an algorithm to embed objects into certain locations in cluttered photographs with a goal to make the objects hard to notice. While these techniques are mostly designed with photographs in mind, our focus is on paintings. In particular, we are interested in the case where the background of the composite is a painting or a drawing.

Style Transfer using Neural Networks. Recent work on Neural Style transfer [GEB16] has shown impressive results on transferring the style of an artwork by matching the statistics of layer responses of a deep neural network. These methods transfer arbitrary styles from one image to another by matching the correlations between feature activations extracted by a pretrained deep neural network on image classification (i.e., VGG [SZ14]). The reconstruction process is based on an iterative optimization framework that minimizes the content and style losses computed from the VGG neural network. Recently, feed-forward generators propose fast approximations of the original Neural Style formulations [ULVL16,JAFF16,LW16b] to achieve real-time performance. However, this technique is sensitive to mismatches in the image content and several approaches have been proposed to address this issue. Gatys et al. [GEB*17] add the possibility for users to guide the transfer with annotations. In the context of photographic transfer, Luan et al. [LPSB17] limit mismatches using scene analysis. Li and Wand [LW16a] use nearest-neighbor correspondences between neural responses to make the transfer content-aware. Specifically, they use a non-parametric model that independently matches the local patches in each layer of the neural network using normalized cross-correlation. Note that this differs from our approach since we use feature representations based on Gram matrices and enforce spatial consistency across different layers in the neural network when computing the correspondence. Improvements can be seen in the comparison results in Section 2.1. Odena et al. [ODO16] study the filters used in these networks and explain how to avoid the grid-like artifacts produced by some techniques. Recent approaches replace the Gram matrix with matching other statistics of neural responses [HB17,LFY*17]. Liao et al. [LYY*17] further improve the quality of the results by introducing bidirectional dense correspondence field matching. All these methods have in common that they change the style of entire images at once. Our work differs in that we focus on local transfer; we shall see that global methods do not work as well when applied locally.

1.2. Background

Our work builds upon the style transfer technique introduced by Gatys et al. [GEB16] (Neural Style) and several additional reconstruction losses proposed later to improve its results. We summarize these techniques below before describing our algorithm in the next section (§ 2).

1.2.1. Style Transfer

Parts of our technique have a similar structure to the Neural Style algorithm by Gatys et al. [GEB16]. They found that recent deep neural networks can learn to extract high-level semantic information and are able to independently manipulate the content and style of natural images. For completeness, we summarize the Neural Style algorithm that transfers a *style* image to an *input* image to produce an output image by minimizing loss functions defined using the VGG network. The algorithm proceeds in three steps:

1. The input image I and style S are processed with the VGG

network [SZ14] to produce a set of activation values as feature representations $F[I]$ and $F[S]$. Intuitively, these capture the statistics that represent the style of each image.

2. The style activations are mapped to the input ones. In the original approach by Gatys et al., the entire set of style activations is used. Other options have been later proposed, e.g., using nearest neighbors neural patches [LW16a].
3. The output image O is reconstructed through an optimization process that seeks to preserve the content of the input image while at the same time match the visual appearance of the style image. These objectives are modeled using *losses* that we describe in more detail in the next section.

Our approach applies this three-step process twice, the main variation being the activation matching step (2). Our first pass uses a matching algorithm designed for robustness to large style differences, and our second pass uses a more constrained matching designed to achieve high visual quality.

1.2.2. Reconstruction Losses

The last step of the pipeline proposed by Gatys et al. is the reconstruction of the final image O. As previously discussed, this involves solving an optimization problem that balances several objectives, each of them modeled by a *loss function*. Originally, Gatys et al. proposed two losses: one to preserve the content of the input image I and one to match the visual appearance of the style image S. Later, more reconstruction losses have been proposed to improve the quality of the output. Our work builds upon several of them that we review below.

Style and Content Losses. In their original work, Gatys et al. used the loss below.

$$\mathcal{L}_{\text{Gatys}} = \mathcal{L}_{\text{c}} + w_{\text{s}}\mathcal{L}_{\text{s}} \tag{1a}$$

$$\text{with: } \mathcal{L}_{\text{c}} = \sum_{\ell=1}^{L} \frac{\alpha_\ell}{2N_\ell D_\ell} \sum_{i=1}^{N_\ell}\sum_{p=1}^{D_\ell} \big(F_\ell[O] - F_\ell[I]\big)^2_{ip} \tag{1b}$$

$$\mathcal{L}_{\text{s}} = \sum_{\ell=1}^{L} \frac{\beta_\ell}{2N_\ell^2} \sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} \big(G_\ell[O] - G_\ell[S]\big)^2_{ij} \tag{1c}$$

where L is the total number of convolutional layers, N_ℓ the number of filters in the ℓ^{th} layer, and D_ℓ the number of activation values in the filters of the ℓ^{th} layer. $F_\ell[\cdot] \in \mathbb{R}^{N_\ell \times D_\ell}$ is a matrix where the (i,p) coefficient is the p^{th} activation of the i^{th} filter of the ℓ^{th} layer and $G_\ell[\cdot] = F_\ell[\cdot]F_\ell[\cdot]^{\mathsf{T}} \in \mathbb{R}^{N_\ell \times N_\ell}$ is the corresponding Gram matrix. α_ℓ and β_ℓ are weights controlling the influence of each layer and w_{s} controls the tradeoff between the *content* (Eq. 1b) and the *style* (Eq. 1c). The advantage of the Gram matrices G_ℓ is that they represent the statistics of the activation values F_ℓ independently of their location in the image, thereby allowing the style statistics to be "redistributed" in the image as needed to fit the input content. Said differently, the product $F_\ell[\cdot]F_\ell[\cdot]^{\mathsf{T}}$ amounts to summing over the entire image, thereby pooling local statistics into a global representation.

Histogram Loss. Wilmot et al. [WRB17] showed that $\mathcal{L}_{\text{Gatys}}$ is unstable because of ambiguities inherent in the Gram matrices and proposed the loss below to ensure that activation histograms are preserved, which remedies the ambiguity.

$$\mathcal{L}_{\text{hist}} = \sum_{\ell=1}^{L} \gamma_\ell \sum_{i=1}^{N_\ell}\sum_{p=1}^{D_\ell} \big(F_\ell[O] - R_\ell[O]\big)^2_{ip} \tag{2a}$$

$$\text{with: } R_\ell[O] = \texttt{histmatch}(F_\ell[O], F_\ell[S]) \tag{2b}$$

where γ_ℓ are weights controlling the influence of each layer and $R_\ell[O]$ is the histogram-remapped feature map by matching $F_\ell[O]$ to $F_\ell[S]$.

Total Variation Loss. Johnson et al. [JAFF16] showed that the total variation loss introduced by Mahendran and Vedaldi [MV15] improves style transfer results by producing smoother outputs.

$$\mathcal{L}_{\text{tv}}(O) = \sum_{x,y} (O_{x,y} - O_{x,y-1})^2 + (O_{x,y} - O_{x-1,y})^2 \tag{3}$$

where the sum is over all the (x,y) pixels of the output image O.

2. Painterly Harmonization Algorithm

We designed a two-pass algorithm to achieve painterly harmonization. Previous work used a single-pass approach; for example, Gatys et al. [GEB16] match the entire style image to the entire input image and then use the L_2 norm on Gram matrices to reconstruct the final result. Li and Wand [LW16a] use nearest neighbors for matching and the L_2 norm on the activation vectors for reconstructing. In our early experiments, we found that such single-pass strategies did not work as well in our context and we were not able to achieve as good results as we hoped. This motivated us to develop a two-pass approach where the first pass aims for coarse harmonization, and the second focuses on fine visual quality (Alg. 1).

The first pass produces an intermediate result that is close to the desired style but we do not seek to produce the highest quality output possible at this point. By relaxing the requirement of high quality, we are able to design a robust algorithm that can cope with vastly different styles. This pass achieves coarse harmonization by first performing a rough match of the color and texture properties of the pasted region to those of semantically similar regions in the painting. We find nearest-neighbor neural patches independently on each network layer (Alg. 3) to match the responses of the pasted region and of the background. This gives us an intermediate result (Fig. 2b) that is a better starting point for the second pass.

Then, in the second pass, we start from this intermediate result and focus on visual quality. Intuitively, since the intermediate image and the style image are visually close, we can impose more stringent requirements on the output quality. In this pass, we work at a single intermediate layer that captures the local texture properties of the image. This generates a correspondence map that we process to remove spatial outliers. We then upsample this spatially consistent map to the finer levels of the network, thereby ensuring that at each output location, the neural responses at all scales come from the same location in the painting (Alg. 4). This leads to more coherent textures and better looking results (Fig. 2c). In the rest of this section, we describe in detail each step of the two passes.

Algorithm 1: *TwoPassHarmonization*(I, M, S)

input input image I and mask M
style image S
output output image O

// Pass #1: Robust coarse harmonization (§ 2.1, Alg. 2)
// Treat each layer independently during input-to-style mapping (Alg. 3)
$I' \leftarrow$ *SinglePassHarmonization*$(I, M, S, IndependentMapping)$

// Pass #2: High-quality refinement (§ 2.2, Alg. 2)
// Enforce consistency across layers
// and in image space during input-to-style mapping (Alg. 4)
$O \leftarrow$ *SinglePassHarmonization*$(I', M, S, ConsistentMapping)$

Algorithm 2: *SinglePassHarmonization*(I, M, S, π)

input input image I and mask M
style image S
neural mapping function π
output output image O

// Process input and style images with VGG network.
$F[I] \leftarrow ComputeNeuralActivations(I)$
$F[S] \leftarrow ComputeNeuralActivations(S)$

// Match each input activation in the mask to a style activation
// and store the mapping from the former to the latter in P.
$P \leftarrow \pi(F[I], M, F[S])$

// Reconstruct output image to approximate new activations.
$O \leftarrow Reconstruct(I, M, S, P)$

2.1. First Pass: Robust Coarse Harmonization

We designed our first pass to be robust to the diversity of paintings that users may provide as style images. In our early experiments, we made two observations. First, we applied the technique of Gatys et al. [GEB16] as is, that is, we used the entire style image to build the style loss $\mathcal{L}_s$. This produced results where the pasted element became a "summary" of the style image. For instance, with Van Gogh's Starry Night, the pasted element had a bit of swirly sky, one shiny star, some of the village structure, and a small part of the wavy trees. While each texture was properly represented, the result was not satisfying because only a subset of them made sense for the pasted element. Then, we experimented with the nearest-neighbor approach of Li and Wand [LW16a]. The intuition is that by assigning the closest style patch to each input patch, it selects style statistics more relevant to the pasted element. Although the generated texture tended to lack contrast compared to the original painting, the results were more satisfying because the texture was more appropriate. Based on these observations, we designed the algorithm below that relies on nearest-neighbor correspondences and a reconstruction loss adapted from [GEB16].

Mapping. Similarly to Li and Wand, for each layer ℓ of the neural network, we stack the activation coefficients at the same location in the different feature maps into an *activation vector*. Instead of considering N_ℓ feature maps, each of them with D_ℓ coefficients, we work with a single map that contains D_ℓ activation vectors of dimension N_ℓ. For each activation vector, we consider the 3×3 patch centered on it. We use nearest neighbors based on the L_2 norm on these patches to assign a style vector to each input vector. We call this strategy *independent mapping* because the assignment is made independently for each layer. Algorithm 3 gives the pseudocode of this mapping. Intuitively, the independence across layers makes the process more robust because a poor match in a layer can be compensated for by better matches in the other layers. The downside of this approach is that the lack of coherence across layers impacts the quality of the output (Fig. 2b). However, as we shall see, these artifacts are limited and our second pass removes them.

(a) *Cut-and-paste*

(b) 1^{st} *pass. Robust harmonization but weak texture (top) and artifacts (bottom).*

(c) 2^{nd} *pass. Refined results with accurate texture and no artifact.*

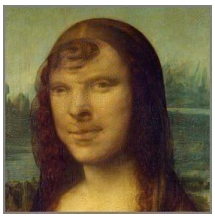
Figure 2: *Starting from vastly different input and style images (a), we first harmonize the overall appearance of the pasted element (b) and then refine the result to finely match the texture and remove artifacts (c).*

Reconstruction. Unlike Li and Wand who use the L_2 norm on these activation vectors to reconstruct the output image, we pool the vectors into Gram matrices and use $\mathcal{L}_{\text{Gatys}}$ (Eq. 1). Applying the L_2 norm directly on the vectors constrains the spatial location of the activation values; the Gram matrices relax this constraint as discussed in § 1.2.2. Figure 3 shows that using L_2 reconstruction directly, i.e., without Gram matrices, does not produce as good results.

2.2. Second Pass: High-Quality Refinement

As can be seen in Figure 2(b), the results after the first pass match the desired style but suffer from artifacts. In our early experiment, we tried to fine-tune the first pass but our attempts only improved some results at the expense of others. Adding constraints to achieve a better quality was making the process less robust to style diversity. We address this challenge with a second pass that focuses on visual quality. The advantage of starting a complete new pass is that we now start from an intermediate image close to the desired

Overly weak texture



Severe artifacts

Figure 3: *Examples of quality loss when not using a Gram matrix in the first pass. The inputs are the same as in Figure 2. Directly applying the L_2 norm, similarly to Li and Wand, forces each spatial location to use only the nearest neighbor from the style image, which causes artifacts when the match is not good. As we show later, using a Gram matrix to pack the matched features together relaxes this limitation and produces fewer artifacts.*

Algorithm 3: *IndependentMapping*$(F[I], M, F[S])$

input input neural activations $F[I]$ and mask M
style neural activations $F[S]$

output input-to-style mapping P

```
// For each layer in the network...
for ℓ ∈ [1 : L] do // L = number of layers
    // For each "activation patch" in the ℓ^th layer...
    //     "activation patch" = vector made of all the activations
    //     in a 3 × 3 patch across all the filters of a layer.
    for p ∈ [1 : D_ℓ] do // D_ℓ = number of patches in the ℓ^th layer
        // Consider only the patches inside the mask
        // resized to the resolution of the ℓ^th layer
        if p ∈ Resize(M, ℓ) then
            // Assign the style patch closest to the input patch
            P(ℓ, p) ← NearestNeighborIndex(F_ℓ[I]_p, F_ℓ[S])
```

result and robustness is not an issue anymore. We design our pass such that the input-to-style mapping is consistent across layers and space. We ensure that the activation vectors assigned to the same image location on different layers were already collocated in the style image. We also favor the configuration where vectors adjacent in the style image remain adjacent in the mapping. Enforcing such strict requirements directly on the input image often yields poor results (Fig. 5d) but when starting from the intermediate image generated by the first pass, this approach produces high quality outputs (Fig. 5h). We also build on previous work to improve the reconstruction step. We explain the details of each step below.

Mapping. We start with a nearest-neighbor assignment similar to the first pass (§ 2.1) but applied only to a single layer, which we call the *reference layer* ℓ_{ref} (Alg. 4, Step #1). We tried several layers for ℓ_{ref} and found that *conv4_1* provided a good trade-off between deeper layers that ignore texture, and shallower layers that ignore scene semantics, e.g., pairing unrelated objects together (Fig. 4).

Then, we process this single-layer mapping to improve its spatial consistency by removing outliers. We favor configurations where

Algorithm 4: *ConsistentMapping*$(F[I], M, F[S])$

input input neural coefficients $F[I]$ and mask M
style neural coefficients $F[S]$

output input-to-style mapping P_{out}

```
// Step #1: Find matches for the reference layer.
// Do the same as in Alg. 3 but only for the reference layer.
for p ∈ [1 : D_ℓref] do
    if p ∈ Resize(M, ℓ_ref) then
        // P is an intermediate input-to-style mapping refined
        // in the next step of the algorithm.
        P(ℓ_ref, p) ← NearestNeighborIndex(F_ℓref[I]_p, F_ℓref[S])

// Step #2: Enforce spatial consistency.
for p ∈ [1 : D_ℓref] do
    if p ∈ Resize(M, ℓ_ref) then
        // Look up the corresponding style patch.
        q ← P(ℓ_ref, p)
        // Initialize a set of candidate style patches.
        CSet ← {q}
        // For all adjacent patches...
        for o ∈ {N, NE, E, SE, S, SW, W, NW} do
            // Duplicate its assignment, i.e.:
            // 1. Look up the style patch of the adjacent patch
            //    p + o
            //    and apply the opposite offset −o.
            // 2. Add the result to the set of candidates.
            CSet ← CSet ∪ {P(ℓ_ref, p + o) − o}
        // Select the candidate the most similar to the style patches
        // associated to the neighbors of p.
        P_out(ℓ_ref, p) ←
            arg min_{c∈CSet} Σ_{o∈O} ||F_ℓref[S]_c − F_ℓref[S]_{P(ℓ_ref, p+o)}||^2
                // with O = {N, NE, E, SE, S, SW, W, NW}

// Step #3: Propagate the matches in the ref. layer to the other layers.
// For each layer in the network excluding the reference layer...
for ℓ ∈ [1 : (ℓ_ref − 1)] ∪ [(ℓ_ref + 1) : L] do
    for p ∈ [1 : D_ℓ] do
        if p ∈ Resize(M, ℓ) then
            // Compute the index of the patch in F_ℓ[I]
            // at the same image location as F_ℓref[I]_p.
            p' ← ChangeResolution[I](ℓ, ℓ_ref, p)
            // Fetch matching style patch in the reference layer.
            q ← P_out(ℓ_ref, p')
            // Change the resolution back.
            P_out(ℓ, p) ← ChangeResolution[S](ℓ_ref, ℓ, q)
```

(a) $\ell_{ref} = \mathit{conv3\_1}$ **(b)** $\ell_{ref} = \mathit{conv4\_1}$ *(our setting)* **(c)** $\ell_{ref} = \mathit{conv5\_1}$

Figure 4: *Setting ℓ_{ref} to conv3_1 produces low-quality results due to poor matches between the input and style images (a). Instead we use conv4_1 that yields better results (b). Using the deeper layer conv5_1 generates lower-quality texture (c) but the degradation is minor compared to using conv3_1. The inputs are the same as in Figure 2.*

all the style vectors assigned to an input region come from the same region in the style image. For each input vector p, we compare the style vector assigned by the nearest-neighbor correspondence above as well as the vectors obtained by duplicating the assignments of the neighbors of p. Among these candidates, we pick the vector pointing to a style feature that is most similar to its neighbors' features. In practice, this removes small outlier regions that are inconsistent with their neighborhood. This procedure is Step #2 of Algorithm 4.

Last, we propagate these correspondences to the other layers so that the activation values are consistent across layers. For a given location in the input image, all the activation values across all the layers are assigned style activations that come from the same location in the style image (Alg. 4, Step #3).

Reconstruction. A first option is to apply the same reconstruction as the first pass (§ 2.1), which already gives satisfying results although some minor defects remain (Fig. 5e). We modify the reconstruction as follows to further improve the output. First, we observe that in some cases, the nearest-neighbor assignment selects the same style vector many times, which generates poor results. We address this issue by selecting each vector at most once and ignoring the additional occurrences, i.e., each vector contributes at most once to the Gram matrix used in the style loss. We name $\mathcal{L}_{s1}$ this variant of the style loss. We also add the histogram and total-variation losses, $\mathcal{L}_{hist}$ and $\mathcal{L}_{tv}$ (§ 1.2.2). Together, these form the loss that we use to reconstruct our final output:

$$\mathcal{L}_{final} = \mathcal{L}_c + w_s\mathcal{L}_{s1} + w_{hist}\mathcal{L}_{hist} + w_{tv}\mathcal{L}_{tv} \quad (4)$$

where the weights w_s, w_{hist}, and w_{tv} control the balance between the terms. Figure 5 illustrates the benefits of this loss. We explain in Section 3 how to set these weights depending on the type of painting provided as the style image.

Discussion. Our constrained mapping was inspired by the nearest-neighbor field upsampling used in the Deep Analogy work [LYY*17, § 4.4] that constrains the matches at each layer to come from a local region around the location at a previous layer. When the input and style images are similar, this technique performs well. In our context, the intermediate image and the style image are even more similar. This encouraged us to be even stricter by forcing the matches to come from the exact same location. Beside this similar aspect, the other algorithmic parts are different and as we shall see, our approach produces better results for our application.

We experimented with using the same reconstruction in the first pass as in the second pass. The quality gains were minimal on the intermediate image and mostly non-existent on the final output. This motivated us to use the simpler reconstruction in the first pass as described in Section 2.1 for the sake of efficiency.

2.3. Post-processing

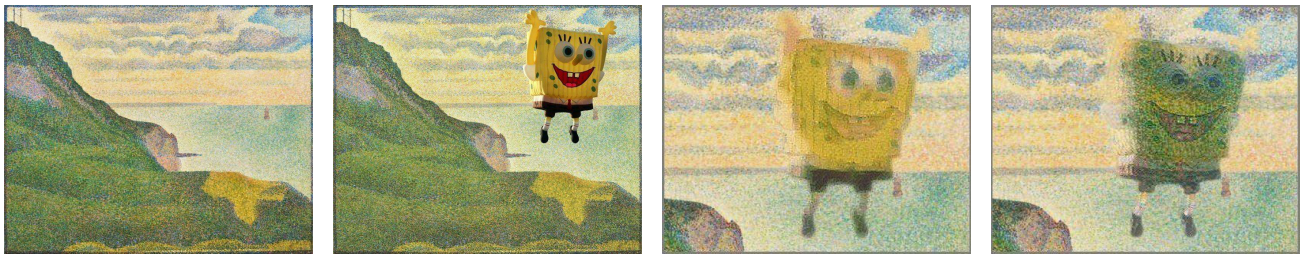
The two-pass process described thus far yields high quality results at medium and large scales but in some cases, fine-scale details can be inaccurate. Said differently, the results are good from a distance but may not be as satisfactory on close examination. The two-step signal-processing approach below addresses this.

Chrominance Denoising. We observed that, in our context, high-frequency artifacts primarily affect the chrominance channels while the luminance is comparatively cleaner. We exploit this characteristic by converting the image to CIE-Lab color space and applying the Guided Filter [HST10] to filter the *ab* chrominance channels with the luminance channel as guide. We use the parameters suggested by the authors, i.e., $r = 2, eps = 0.1^2$. This effectively suppresses the highest-frequency color artifacts. However, some larger defects may remain. The next step addresses this issue.

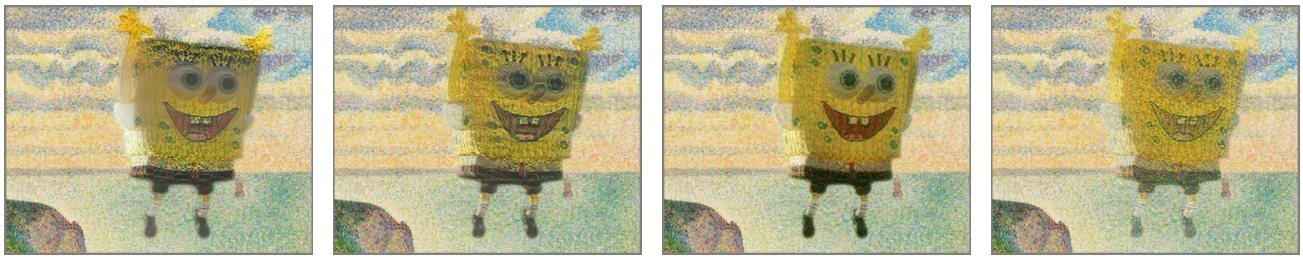
Patch Synthesis. The last step uses patch synthesis to ensure that every image patch in the output appears in the painting. We use PatchMatch [BSFG09] to find a similar style patch to each output patch. We reconstruct the output by averaging all overlapping style patches, thereby ensuring that no new content is introduced. However, the last averaging step tends to smooth details. We mitigate this effect by separating the image into a *base layer* and a *detail layer* using the Guided Filter again (using the same parameters). The base layer is the output of the filter and contains the coarse image structure, and the detail layer is the difference with the original image that contains the high-frequency details. We then apply patch synthesis on the base layer only and add back the details. This procedure ensures that the texture is not degraded by the averaging, thereby producing crisp results.

3. Painting Estimator

The above algorithm has two important parameters that affect the stylistic properties of the output — style and histogram weights (w_s and w_{hist}). We observed that different sets of parameters gave optimal results for different paintings based on their level of stylization. For example, Cubism paintings often contain small multifaceted areas with strong and sharp brush strokes, while High Renaissance



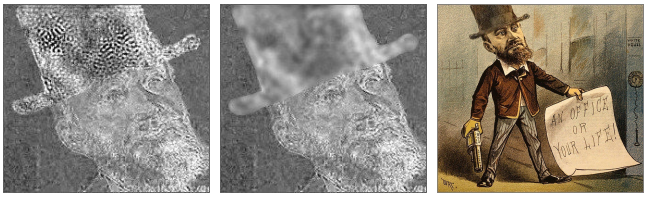
(a) *Style image* **(b)** *Cut-and-paste* **(c)** *Independent mapping (1^{st} pass only, our intermediate result)* **(d)** *Consistent mapping (2^{nd} pass only, bad correspondence)*



(e) *Entire pipeline without $\mathcal{L}_{hist}$ and using $\mathcal{L}_s$ instead of $\mathcal{L}_{s1}$* **(f)** *Entire pipeline using $\mathcal{L}_s$ instead of $\mathcal{L}_{s1}$* **(g)** *Entire pipeline without painting estimator (default parameters, style is too weak)* **(h)** *Our final result*

Figure 5: *Ablation study. (c-h) are cropped for best fit. Zoom in for details. The 1^{st} pass (a) reduces the style difference gap between the foreground and background, but lacks fine texture details. Directly applying the 2^{nd} pass produces fine texture details, but the correspondence is not as good, which causes texture mismatch problems. Without the histogram loss, (e) is unable to reproduce the dot texture of the background painting in some regions due to the instabilities of the Gram matrix. Without addressing many-to-one mappings, redundant style patches are reused multiple times and cause artifacts in the output (f). Without the painting estimator, the style of the result (g) is not well reproduced. (h) is our final solution by combining all components.*

(a) *Inset of our result (before post-processing)* **(b)** *After guided filtering* **(c)** *After patch synthesis*



(d) *a channel (before guided filtering)* **(e)** *a channel (after guided filtering)* **(f)** *Our result after post-processing*

Figure 6: *Post-processing: Given deconvolution result with inset (a), we perform* Chrominance Denoising *to produce (b) and* Patch Synthesis *on (b) to produce (c). (d) and (e) show the insets of a channel in CIE-Lab space before and after denosing. (f) is the final full-resolution result.*

Strength	Art style examples	(w_s, w_{hist})
Weak	Baroque, High Renaissance	$(1.0, 1.0)$
Medium	Abstract Art, Post-Impressionism	$(5.0, 5.0)$
Strong	Cubism, Expressionism	$(10.0, 10.0)$

Table 1: *Weights for selected art styles. Please refer to the supplemental document for compact art styles and parameter weights. The final weight is a linear interpolation of different art styles using our trained painting estimator network. TV weights are computed separately based on the noise level of the painting image (Sec. 4) .*

and Baroque paintings are more photorealistic. Rather than tweak parameters for each input, we developed a trained predictor of the weights to make our approach to weight selection more robust.

We train a *painting estimator* that predicts the optimization parameters for our algorithm such that parameters that allow deeper style changes are used when the background painting is more stylized and vice versa. To train this estimator, we split the parameter values into three categories ("Weak", "Medium" and "Strong"), and manually assign each painting style to one of the categories. Table 1 presents a subset of painting styles and their categories and weight values. We selected the 18 most common styles based on `wikiart.org` ranking: Abstract Art, Abstract Expressionism, Art Nouveau (Modern), Baroque, Color Field Painting, Cu-

bism, Early Renaissance, Expressionism, High Renaissance, Impressionism, Mannerism (Late Renaissance), Naive Art (Primitivism), Northern Renaissance, Post-Impressionism, Realism, Surrealism, Symbolism and Ukiyo-e. More details appear in the supplementary material.

We collected 80,000 paintings from `wikiart.org` and fine-tuned the VGG-16 network [SZ14] on classifying 18 different styles. After training, we remove the last classification layer and use weighted linear interpolation on the softmax layer based on style categories to output a floating value for w_s and w_{hist} indicating the level of stylization. These parameter values (shown in Table 1) are then used in the optimization.

4. Implementation Details

This section describes the implementation details of our approach. We employed pre-trained VGG-19 [SZ14] as the feature extractor. For the first-pass optimization, we chose *conv4_1* ($\alpha_\ell = 1$ for this layer and $\alpha_\ell = 0$ for all other layers) as the content representation, and *conv3_1*, *conv4_1* and *conv5_1* ($\beta_\ell = 1/3$ for those layers and $\beta_\ell = 0$ for all other layers) as the style representation, since higher layers have been transformed by the CNN into representations with more of the actual content, which is crucial for the semantic-aware nearest-neighbor search. We used these layer preferences for all the first-pass results. For the second-pass optimization, we chose *conv4_1* as the content representation, *conv1_1*, *conv2_1*, *conv3_1* and *conv4_1* as the style representation. We also employed the histogram loss and used *conv1_1*, and *conv4_1* ($\gamma_\ell = 1/2$ for these layers and $\gamma_\ell = 0$ for all other layers) as the histogram representation as suggested by the original authors. We chose *conv4_1* as the reference layer ℓ_{ref} for the nearest-neighbor search in the second-pass optimization. We name τ the output floating number of the painting estimator and set the parameters $w_s = \tau$, $w_{\text{hist}} = \tau$, and $w_{\text{tv}} = \tau * \texttt{sigmoid}(\texttt{median\_tv}(S))$, where the $\texttt{sigmoid}(x) = \frac{10}{1+\exp(10^4 x - 25)}$ and $\texttt{median\_tv}(S)$ is the median total variation (Eq. 3) of the painting S. We found the parameters for the sigmoid function empirically. The intuition is that we impose less smoothness when the original painting is textured.

Our main algorithm is developed in Torch + CUDA. We have implemented the dense correspondence search and spatial consistency resampling in CUDA to manipulate the feature activations from the VGG network. The optimization pipeline is based on a popular Torch implementation [†]. We apply chrominance denoising and patch synthesis after the optimization pipeline as a separate post-processing step. All our experiments are conducted on a PC with an Intel Xeon E5-2686 v4 processor and an NVIDIA Tesla K80 GPU. We use the L-BFGS solver [LN89] for the reconstruction with 1000 iterations. The runtime on a 500×500 image takes about 5 minutes. We will release our implementation upon acceptance for non-commercial use and future research.

5. Results

We now evaluate our harmonization algorithm in comparison with related work and through user studies.

† https://github.com/jcjohnson/neural-style

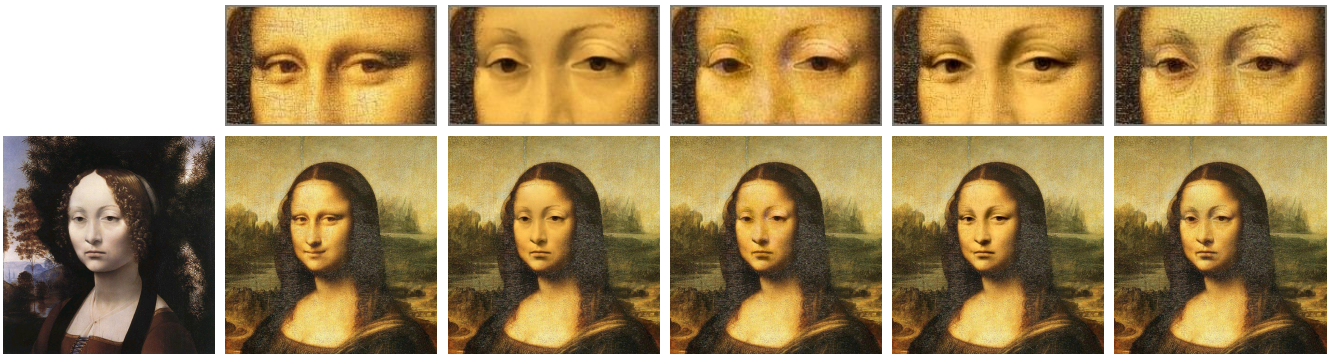
Main Results. In Figures 11 and 12, we compare our method with four state-of-the-art methods: Neural Style [GEB16], CNNMRF [LW16a], Multi-Scale Image Harmonization [SJMP10], and Deep Image Analogy [LYY*17] across paintings with various styles and genres. Neural Style tends to produce "style summaries" that rarely work well; for example, background sky texture appears in the foreground eiffel tower (Fig. 11(iv)). This is due to the lack of semantic matching since the Gram matrix is computed over the entire painting. CNNMRF often generates weak style transfers that do not look as good when juxtaposed with the original painting. Multi-Scale Image Harmonization performs noise matching to fit high-frequency inter-scale texture but does not capture spatially-varying brush strokes common in paintings with heavy styles. Deep Image Analogy is more robust compared to the other three methods but its results are sometimes blurred due to patch synthesis and voting, e.g., Figures 11(i-iii), 12(vii). Its coarse-to-fine pipeline also sometimes misses parts (Fig. 11(iv)).

User Studies. We conduct user studies to quantitatively characterize the quality of our results. The first user study, "Edited or Not", aims to understand whether the harmonization quality is good enough to fool an observer. The second user study, "Comparison", compares the quality of our results with that of related algorithms.

Study 1: Edited or Not. We showed the users 20 painterly composites, each edited by one of four algorithms: CNNMRF [LW16a], Multi-Scale Image Harmonization [SJMP10], Deep Image Analogy [LYY*17], and ours. We asked the users whether the painting had been edited. If they thought it was, we asked them to click on the part of the image they believed was edited (this records the coordinates of the edited object) so that we could verify the correctness of the answer. This verification is motivated by our pilot study where we found that people would sometimes claim an image is edited by erroneously identifying an element as edited although it was part of the original painting. Such misguided classification is actually a positive result that shows the harmonization is of high quality. We also recorded the time it took users to answer each question.

One potential problem in the study that we had to consider is that people might spot the edited object in a painting due to reasons other than the harmonization quality. For example, an edit in a famous painting will be instantly recognizable, or if the composition of the edited painting is semantically wrong, e.g., a man's face in a woman's head, or a spaceship in a painting from the 19th century. To avoid these problems, we selected typically unfamilar paintings and made the composition sensible; for example adding a park bench in a meadow or a clock on a wall (see supplementary material for more examples). We further asked the users for each example if they were familiar with the painting. If they were, we eliminated their judgement as being tainted by prior knowledge.

Figure 8 shows the results of Study 1 using two metrics: the average *painting classification rate* and average *answer time*. Let $N(x)$ denote the total number of users with answer x. For an original painting, the painting classification rate is computed as $N_n/(N_n + N_e)$, where N_n is the number of answers with `Not Edited` and N_e is the number of answers with `Edited`. For edited paintings using those four algorithms, the painting classification rate P is computed as $(N_n + \hat{N}_e)/(N_n + N_e)$, where $\hat{N}_e$ is the number of answers with



(a) *Source* **(b)** *Target painting* **(c)** *[PGB03]* **(d)** *[SJMP10]* **(e)** *[DSB*12]* **(f)** *Ours*

Figure 7: *When pasting the face of Ginevra de' Benci (a) on Mona Lisa (b), Poisson Blending [PGB03] does not match the texture (c), Mulitscale Harmonization [SJMP10] adds texture but does not reproduce the paint cracks (d), Image Melding [DSB*12] adds cracks faithfully but not everywhere, e.g., there are no cracks below the eye on the right (e). In comparison, our result generates cracks more consistently over the image (f).*

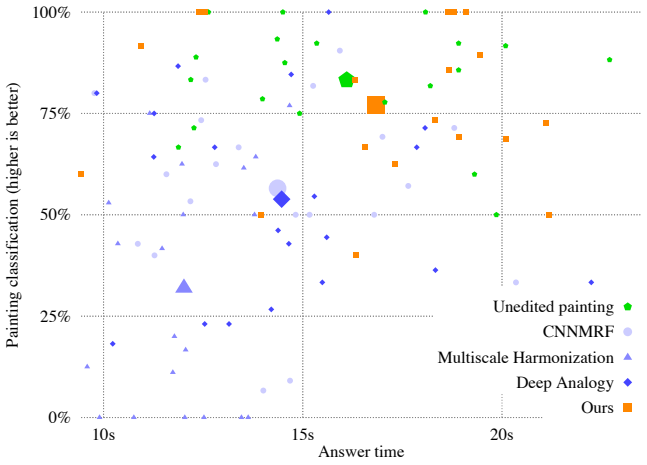


Figure 8: *Results of the "Edited or Not" user study. A higher painting classification rate means better harmonization performance since users were unable to identify the edit. See text for more details. The large symbols represent the average of each category.*

`Edited` but with the wrong XY coordinates of the mouse click. This captures all the cases where the viewer was "fooled" by the harmonization result. A higher rate means better harmonization quality since users were unable to identify the modification. Figure 8 shows that our algorithm achieves a painting classification rate significantly higher than that of the other algorithms and close to that of unedited paintings.

The answer time has a less straightforward interpretation since it may also reflect how meticulous users are. Nonetheless, Figure 8 shows that the answer time for our algorithm is close to that of unedited paintings and significantly different from that of the other algorithms, which also suggests that our results share more similarities with actual paintings than the outputs of the other methods.

Study 2: Comparison. We showed the users 17 paintings, each painting had been edited with the same four algorithms as the first

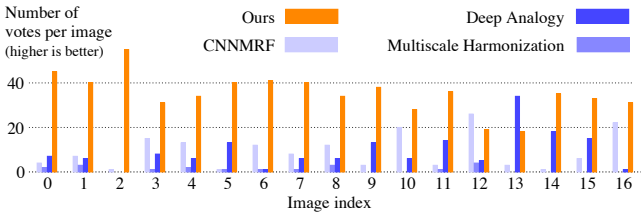


Figure 9: *Results of the "Comparison" user study. Our algorithm is often the most preferred among the four algorithms.*

study. We asked the users to select the result that best captures the consistency of the colors and of the texture in the painting. The quantitative results are shown in Figure 9. For most paintings our algorithm is most preferred. We provide more detailed results in the supplementary document.

Image Harmonization Comparisons. We compare our results with Poisson blending and two state-of-the-art harmonization solutions, [SJMP10] and [DSB*12] in Figure 7. Poisson blending achieves good overall color matching but does not capture the texture of the original painting. Multiscale Harmonization [SJMP10] transforms noise to transfer texture properties, in addition to color and intensity. However, it is designed to fit small-scale, noise-like texture and is not well suited for more structured patterns such as painting brush strokes and cracks. Image Melding [DSB*12] improves texture quality by using patch synthesis combined with Poisson blending, but the texture disappears at some places. In comparison, our method better captures both the spatial and inter-scale texture and structure properties (Fig. 7f).

Harmonization of a canonical object across styles. In the above examples, we picked different objects for different paintings to create plausible combinations. In this experiment, we use the same canonical object across a variety of styles to demonstrate the stylization independent of the inserted object. We introduce a hot air balloon into paintings with a wide range of styles. We randomly selected paintings from the `wikiart.org` dataset to paste the balloon into (Fig. 10). Note that some of the results are stylized so

strongly that they are difficult to distinguish from the background painting. This is similiar to Camouflage Images [CHM*10], and is a limitation of our approach and an interesting future research direction.

Figure 10: *Canonical object harmonization results for hot air balloon (upper-left).*

6. Conclusions

We have described an algorithm to copy an object in a photograph and paste it into a painting seamlessly, i.e., the composite still looks like a genuine painting. We have introduced a two-pass algorithm that first transfers the overall style of the painting to the input and then refines the result to accurately match the painting's color and texture. This latter pass relies on mapping neural response statistics that ensures consistency across the network layers and in image space. To cope with different painting styles, we have trained a separate network to adjust the transfer parameters as a function of the style of the background painting. Our experiments show that our approach succeeds on a diversity of input and style images, many of which are challenging for other methods. We have also conducted two user studies that show that users often identify our results as unedited paintings and prefer them to the outputs of other techniques.

We believe that our work opens new possibilities for creatively editing and combining images and hope that it will inspire artists. From a technical perspective, we have demonstrated that global painterly style transfer methods are not well suited for local transfer, and we designed an effective local approach. This suggests fundamental differences between the local and global statistics of paintings, and further exploring this difference is an exciting avenue for future work. Other avenues of future work include fast feed-forward network approximations of our optimization framework, as well an extension to painterly video compositing.

Acknowledgements. This work was supported by a National Science Foundation grant (IIS-1617861), Adobe, and a Google Faculty Research Award.

References

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. 28*, 3 (2009), 24–1. 2, 6

[CHM*10] CHU H.-K., HSU W.-H., MITRA N. J., COHEN-OR D., WONG T.-T., LEE T.-Y.: Camouflage images. *ACM Trans. Graph. 29*, 4 (2010), 51–1. 2, 10

[CKIW15] CHEN Z., KIM B., ITO D., WANG H.: Wetbrush: Gpu-based 3d painting simulation at the bristle level. *ACM Trans. Graph. 34*, 6 (2015), 200. 1

[DSB*12] DARABI S., SHECHTMAN E., BARNES C., GOLDMAN D. B., SEN P.: Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph. 31*, 4 (2012), 82–1. 2, 9

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). 2, 3, 4, 8, 11, 12

[GEB*17] GATYS L. A., ECKER A. S., BETHGE M., HERTZMANN A., SHECHTMAN E.: Controlling perceptual factors in neural style transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 2

[HB17] HUANG X., BELONGIE S.: Arbitrary style transfer in real-time with adaptive instance normalization. In *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2017). 2

[HST10] HE K., SUN J., TANG X.: Guided image filtering. In *Proceedings of European Conference on Computer Vision (ECCV)* (2010), Springer-Verlag, pp. 1–14. 6

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision (ECCV)* (2016), Springer, pp. 694–711. 2, 3

[LFY*17] LI Y., FANG C., YANG J., WANG Z., LU X., YANG M.-H.: Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems* (2017). 2

[LN89] LIU D. C., NOCEDAL J.: On the limited memory bfgs method for large scale optimization. *Mathematical programming 45*, 1 (1989), 503–528. 8

[LPSB17] LUAN F., PARIS S., SHECHTMAN E., BALA K.: Deep photo style transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 2

[LW16a] LI C., WAND M.: Combining markov random fields and convolutional neural networks for image synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). 2, 3, 4, 8, 11, 12

[LW16b] LI C., WAND M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of European Conference on Computer Vision (ECCV)* (2016), Springer, pp. 702–716. 2

[LYY*17] LIAO J., YAO Y., YUAN L., HUA G., KANG S. B.: Visual attribute transfer through deep image analogy. *ACM Trans. Graph. 36*, 4 (2017), 120:1–120:15. 1, 2, 6, 8, 11, 12

[MV15] MAHENDRAN A., VEDALDI A.: Understanding deep image representations by inverting them. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 5188–5196. 3

[ODO16] ODENA A., DUMOULIN V., OLAH C.: Deconvolution and checkerboard artifacts. *Distill 1*, 10 (2016), e3. 2

[PD84] PORTER T., DUFF T.: Compositing digital images. In *ACM Siggraph Computer Graphics* (1984), vol. 18, ACM, pp. 253–259. 2

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. In *ACM Trans. Graph.* (2003), vol. 22, ACM, pp. 313–318. 1, 2, 9

[SJMP10] SUNKAVALLI K., JOHNSON M. K., MATUSIK W., PFISTER H.: Multi-scale image harmonization. In *ACM Trans. Graph.* (2010), vol. 29, ACM, p. 125. 1, 2, 8, 9, 11, 12

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014). 2, 3, 8

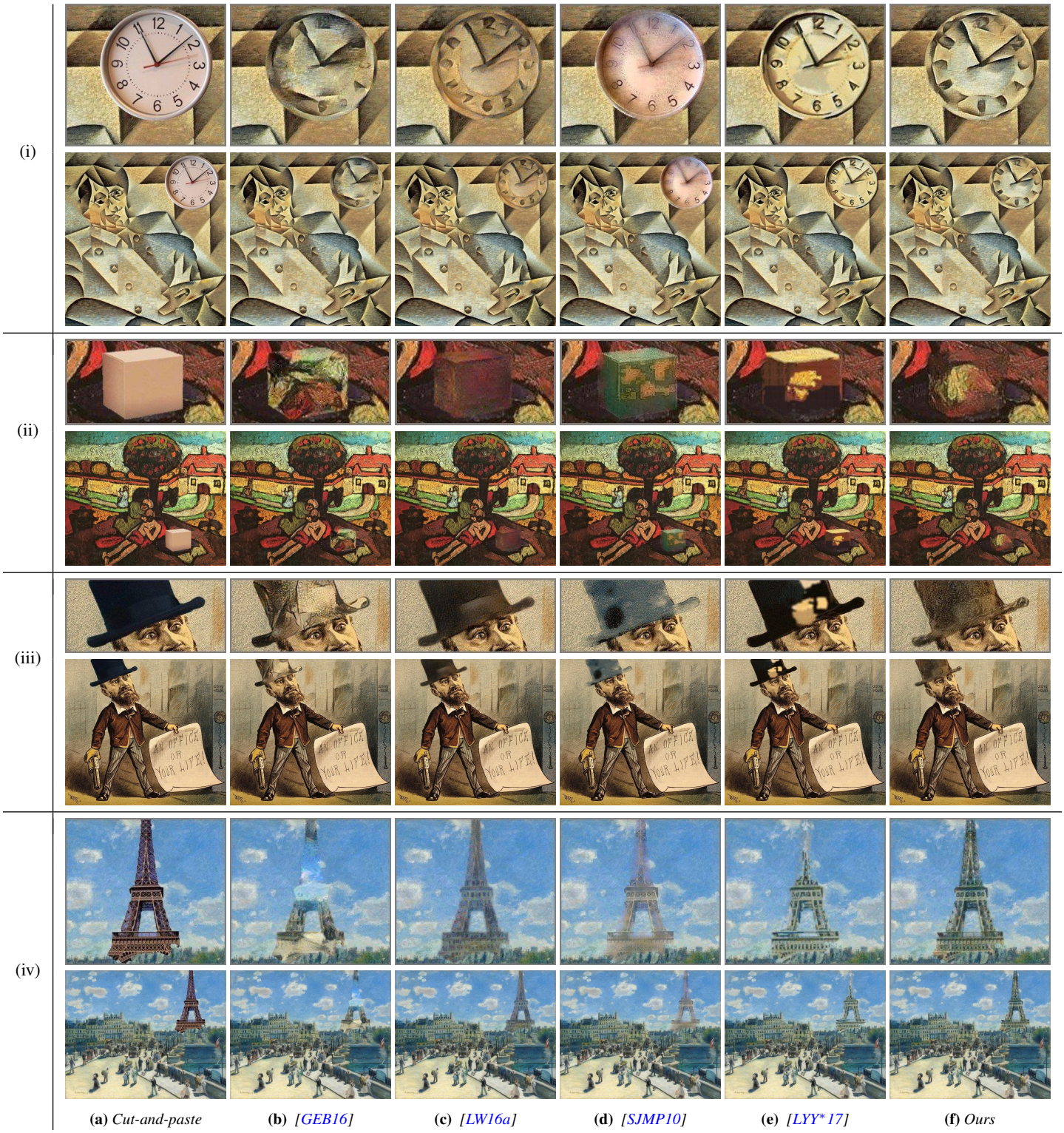


Figure 11: *Example results with insets on proposed composite for unadjusted cut-and-paste, four state-of-the-art methods and our results. We show that our method captures both spatial and inter-scale color and texture and produces harmonized results on paintings with various styles. Zoom in for details.*

Figure 12: *Continued.*

[TSL*17] TSAI Y.-H., SHEN X., LIN Z., SUNKAVALLI K., LU X., YANG M.-H.: Deep image harmonization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 2

[ULVL16] ULYANOV D., LEBEDEV V., VEDALDI A., LEMPITSKY V.: Texture networks: Feed-forward synthesis of textures and stylized images. In *Proceedings of International Conference on Machine Learning* (2016), JMLR.org, pp. 1349–1357. 2

[WRB17] WILMOT P., RISSER E., BARNES C.: Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893* (2017). 3

[XADR12] XUE S., AGARWALA A., DORSEY J., RUSHMEIER H.: Understanding and improving the realism of image composites. *ACM Trans. Graph. 31*, 4 (2012), 84. 1, 2

[ZKSE15] ZHU J.-Y., KRAHENBUHL P., SHECHTMAN E., EFROS A. A.: Learning a discriminative model for the perception of realism in composite images. In *The IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 3943–3951. 2