Generative Visual Manipulation on the Natural Image Manifold

Jun-Yan Zhu[1(✉)], Philipp Krähenbühl[1], Eli Shechtman[2], and Alexei A. Efros[1]

[1] University of California, Berkeley, USA
{junyanz,philkr,efros}@eecs.berkeley.edu
[2] Adobe Research, San Jose, USA
elishe@adobe.com

Abstract. Realistic image manipulation is challenging because it requires modifying the image appearance in a user-controlled way, while preserving the realism of the result. Unless the user has considerable artistic skill, it is easy to "fall off" the manifold of natural images while editing. In this paper, we propose to learn the natural image manifold directly from data using a generative adversarial neural network. We then define a class of image editing operations, and constrain their output to lie on that learned manifold at all times. The model automatically adjusts the output keeping all edits as realistic as possible. All our manipulations are expressed in terms of constrained optimization and are applied in near-real time. We evaluate our algorithm on the task of realistic photo manipulation of shape and color. The presented method can further be used for changing one image to look like the other, as well as generating novel imagery from scratch based on user's scribbles.

1 Introduction

Today, visual communication is sadly one-sided. We all perceive information in the visual form (through photographs, paintings, sculpture, etc.), but only a chosen few are talented enough to effectively express themselves visually. This imbalance manifests itself even in the most mundane tasks. Consider an online shopping scenario: a user looking for shoes has found a pair that mostly suits her but she would like them to be a little taller, or wider, or in a different color. How can she communicate her preference to the shopping website? If the user is also an artist, then a few minutes with an image editing program will allow her to transform the shoe into what she wants, and then use image-based search to find it. However, for most of us, even a simple image manipulation in Photoshop presents insurmountable difficulties. One reason is the lack of "safety wheels" in image editing: any less-than-perfect edit immediately makes the image look completely unrealistic. To put another way, classic visual manipulation paradigm does not prevent the user from "falling off" the manifold of natural images.

Understanding and modeling the natural image manifold has been a long-standing open research problem. But in the last two years, there has been rapid

B. Leibe et al. (Eds.): ECCV 2016, Part V, LNCS 9909, pp. 597–613, 2016.
DOI: 10.1007/978-3-319-46454-1_36

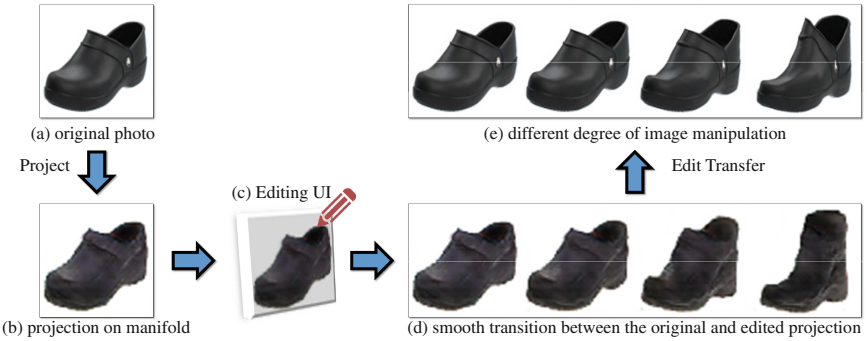

Fig. 1. We use generative adversarial networks (GAN) [1,2] to perform image editing on the natural image manifold. We first project an original photo (a) onto a low-dimensional latent vector representation (b) by regenerating it using GAN. We then modify the color and shape of the generated image (d) using various brush tools (c) (for example, dragging the top of the shoe). Finally, we apply the same amount of geometric and color changes to the original photo to achieve the final result (e). **See interactive image editing video on our website.**

advancement, fueled largely by the development of the generative adversarial networks [1]. In particular, several recent papers [1–5] have shown visually impressive results sampling random images drawn from the natural image manifold. However, there are two reasons preventing these advances from being useful in practical applications at this time. First, the generated images, while good, are still not quite photo-realistic (plus there are practical issues in making them high resolution). Second, these generative models are setup to produce images by sampling a latent vector-space, typically at random. So, these methods are not able to create and/or manipulate visual content in a user-controlled fashion.

In this paper, we use the generative adversarial neural network to learn the manifold of natural images, but we do not actually employ it for image generation. Instead, we use it as a constraint on the output of various image manipulation operations, to make sure the results lie on the learned manifold at all times. This enables us to reformulate several editing operations, specifically color and shape manipulations, in a natural and data-driven way. The model automatically adjusts the output keeping all edits as realistic as possible (Fig. 1).

We show three applications based on our system: (1) Manipulating an existing photo based on an underlying generative model to achieve a different look (shape and color); (2) "Generative transformation" of one image to look more like another; (3) Generate a new image from scratch based on user's scribbles and warping UI.

All manipulations are performed in a straightforward manner through gradient-based optimization, resulting in a simple and fast image editing tool. We hope that this work inspires further research in data-driven generative image editing, and thus release the code and data at our website.

2 Prior Work

Image editing and user interaction: Image editing is a well established area in computer graphics where an input image is manipulated to achieve a certain goal specified by the user. Examples of basic editing include changing the color properties of an image either globally [6] or locally [7]. More advanced editing methods such as image warping [8,9] or structured image editing [10] intelligently reshuffle the pixels in an image following user's edits. While achieving impressive results in the hands of an expert, when these types of methods fail, they produce results that look nothing like a real image. Common artifacts include unrealistic colors, exaggerated stretching, obvious repetitions and over-smoothing. This is because they rely on low-level principles (e.g., similarity of color, gradients or patches) and do not capture higher-level information about natural images.

Image morphing: There are a number of techniques for producing a smooth visual transition between two input images. Traditional morphing methods [11] combine an intensity blend with a geometric warp that requires a dense correspondence. In Regenerative Morphing [12] the output sequence is regenerated from small patches sampled from the source images. Thus, each frame is constrained to look similar to the two sources. Exploring Photobios [13] presented an alternative way to transition between images, by finding a shortest path in a large image collection based on pairwise image distances. Here we extend this idea and produce a morph that is both close to the two sources and stays on, or close to, the natural image manifold.

Natural image statistics: Generative models of local image statistics have long been used as a prior for image restoration problems such as image denoising and deblurring. A common strategy is to learn local filter or patch models, such as Principal Components, Independent Components, Mixture of Gaussians or wavelet bases [14–16]. Some methods attempt to capture full-image likelihoods [17] through dense patch overlap, though the basic building block is still small patches that do not capture global image structures and long range relations. Zhu et al. [18] recently showed that discriminative deep neural networks learn a much stronger prior that captures both low-level statistics, as well as higher order semantic or color-balance clues. This deep prior can be directly used for a limited set of editing operations (e.g. compositing). However it does not extend to the diversity of editing operations considered in this work.

Neural generative models: There is a large body of work on neural network based models for image generation. Early classes of probabilistic models of images include restricted Boltzmann machines (e.g., [19]) and their deep variants [20], auto-encoders [19,21] and more recently, stochastic neural networks [3,22,23] and deterministic networks [24]. Generative adversarial networks (GAN), proposed by Goodfellow et al. [1], learn a generative network jointly with a second discriminative adversarial network in a mini-max objective. The discriminator tries to distinguish between the generated samples and natural image samples, while the generator tries to *fool* the discriminator producing highly realistic looking images.

(a) random samples (b) random jittering (c) linear interpolation

Fig. 2. GAN as a manifold approximation. (a) Randomly generated examples from a GAN, trained on the shirts dataset; (b) random jittering: each row shows a random sample from a GAN (the first one at the left), and its variants produced by adding Gaussian noise to z in the latent space; (c) interpolation: each row shows two randomly generated images (first and last), and their smooth interpolations in the latent space.

Unfortunately in practice, GAN does not yield a stable training objective, so several modifications have been proposed recently, such as a multi-scale generation [4] and a convolution-deconvolution architecture with batch normalization [2]. While the above methods attempt to generate an image starting from a random vector, they do not provide tools to change the generation process with intuitive user controls. In this paper we try to remedy this by learning a generative model that can be easily controlled via a few intuitive user edits.

3 Learning the Natural Image Manifold

Let us assume that all natural images lie on an ideal low-dimensional manifold $\mathbb{M}$ with a distance function $S(x_1, x_2)$ that measures the perceptual similarity between two images $x_1, x_2 \in \mathbb{M}$. Directly modeling this ideal manifold $\mathbb{M}$ is extremely challenging, as it involves training a generative model in a highly structured and complex million dimensional space. Following the recent success of deep generative networks in generating natural looking images, we approximate the image manifold by learning a model using generative adversarial networks (GAN) [1,2] from a large-scale image collection. Beside the high quality results, GAN has a few other useful properties for our task we will discuss next.

Generative Adversarial Networks: A GAN model consists of two neural networks: (1) a generative network $G(z; \theta_g)$ that generates an image $x \in \mathbb{R}^{H \times W \times C}$ given a random vector $z \in \mathbb{Z}$, where $\mathbb{Z}$ denotes a d-dimensional latent space, and (2) a discriminative network $D(x; \theta_d)$ that predicts a probability of a photo being real ($D = 1$) or generated ($D = 0$). For simplicity, we denote $G(z; \theta_G)$ and $D(x; \theta_D)$ as $G(z)$ and $D(x)$ in later sections. One common choice of $\mathbb{Z}$ is a multivariate uniform distribution $Unif[-1, 1]^d$. D and G are learned using a min-max objective [1]. GAN works well when trained on images of a certain class. We formally define $\tilde{\mathbb{M}} = \{G(z) | z \in \mathbb{Z}\}$ and use it as an approximation to the ideal manifold $\mathbb{M}$ (i.e. $\tilde{\mathbb{M}} \approx \mathbb{M}$). We also approximate the distance function of two generated images as an Euclidean distance between their corresponding latent vectors, i.e., $S(G(z_1), G(z_2)) \approx \|z_1 - z_2\|^2$.

GAN as a manifold approximation: We use GAN to approximate an ideal manifold for two reasons: first, it produces high-quality samples (see Fig. 2(a) for example). Though lacking visual details sometimes, the model can synthesize appealing samples with a plausible overall structure. Second, the Euclidean distance in the latent space often corresponds to a perceptually meaningful visual similarity (see Fig. 2(b) for examples). We therefore argue that GAN is a powerful generative model for modeling the image manifold.

Traversing the manifold: Given two images on the manifold $G(z_0), G(z_N)) \in \tilde{\mathbb{M}}$, one would like to seek a sequence of $N+1$ images $\left[G(z_0), G(z_1), \ldots G(z_N)\right]$ with a smooth transition. This is often done by constructing an image graph with images as nodes, and pairwise distance function as the edge, and computing a shortest path between the starting image and end image [13]. In our case, we minimize $\sum_{t=0}^{N-1} S(G(z_t), G(z_{t+1}))$ where S is the distance function. In our case $S(G(z_1), G(z_2)) \approx \|z_1 - z_2\|^2$, so a simple linear interpolation $\left[(1-\frac{t}{N}) \cdot z_0 + \frac{t}{N} \cdot z_N\right]_{t=0}^{N}$ is the shortest path. Figure 2(c) shows a smooth and meaningful image sequence generated by interpolating between two points in the latent space. We will now use this approximation of the manifold of natural images for realistic photo editing.

4 Approach

Figure 1 illustrates the overview of our approach. Given a real photo, we first project it onto our approximation of the image manifold by finding the closest latent feature vector z of the GAN to the original image. Then, we present a real-time method for gradually and smoothly updating the latent vector z so that it generates a desired image that both satisfies the user's edits (e.g. a scribble or a warp; more details in Sect. 5) and stays close to the natural image manifold. Unfortunately, in this transformation the generative model usually looses some of the important low-level details of the input image. We therefore propose a dense correspondence method that estimates both per-pixel color and shape changes from the edits applied to the generative model. We then transfer these changes to the original photo using an edge-aware interpolation technique and produce the final manipulated result.

4.1 Projecting an Image onto the Manifold

A real photo x^R lies, by definition, on the ideal image manifold $\mathbb{M}$. However for an approximate manifold $\tilde{\mathbb{M}}$, our goal here is to find a generated image $x^* \in \tilde{\mathbb{M}}$ close to x^R in some distance metric $\mathcal{L}(x_1, x_2)$ as

$$x^* = \arg\min_{x \in \tilde{\mathbb{M}}} \mathcal{L}(x,\, x^R). \tag{1}$$

For the GAN manifold $\tilde{\mathbb{M}}$ we can rewrite the above equation as follows:

$$z^* = \arg\min_{z \in \tilde{\mathbb{Z}}} \mathcal{L}(G(z),\, x^R). \tag{2}$$

Our goal is to reconstruct the original photo x^R using the generative model G by minimizing the reconstruction error, where $\mathcal{L}(x_1, x_2) = \|\mathcal{C}(x_1) - \mathcal{C}(x_2)\|^2$ in some differentiable feature space $\mathcal{C}$. If $\mathcal{C}(x) = x$, then the reconstruction error is simply pixel-wise Euclidean error. Previous work [5,25] suggests that using deep neural network activations leads to a reconstruction of perceptually meaningful details. We found that a weighted combination of raw pixels and *conv4* features ($\times 0.002$) extracted from AlexNet [26] trained on ImageNet [27] to perform best.

Projection via optimization: As both the feature extractor $\mathcal{C}$ and the generative model G are differentiable, we can directly optimize the above objective using L-BFGS-B [28]. However, the cascade of $\mathcal{C}(G(z))$ makes the problem highly non-convex, and as a result, the reconstruction quality strongly relies on a good initialization of z. We can start from multiple random initializations and output the solution with the minimal cost. However the number of random initializations required to obtain a stable reconstruction is prohibitively large (more than 100), which makes real-time processing impossible. We instead train a deep neural network to minimize Eq. 2 directly.

Projection via a feedforward network: We train a feedforward neural network $P(x; \theta_P)$ that directly predicts the latent vector z from a x. The training objective for the predictive model P is written as follows:

$$\theta_P^* = \arg\min_{\theta_P} \sum_n \mathcal{L}(G(P(x_n^R; \theta_P)),\, x_n^R), \tag{3}$$

where x_n^R denotes the n-th image in the dataset. The architecture of the model P is equivalent to the discriminator D of the adversarial networks, and only varies in the final number of network outputs. Objective 3 is reminiscent of an auto-encoder pipeline, with a encoder P and decoder G. However, the decoder G is fixed throughout the training. While the optimization problem 2 is exactly the same as the learning objective 3, the learning based approach often performs better and does not fall into local optima. We attribute this behavior to the regularity in the projection problem and the limited capacity of the network P. Projections of similar images will share similar network parameters and produce a similar result. In some sense the loss for one image provides information for many more images that share a similar appearance [29]. However, the learned inversion is not always perfect, and can often be improved further by a few additional steps of optimization.

A hybrid method: The hybrid method takes advantage of both approaches above. Given a real photo x^R, we first predict $P(x^R; \theta_P)$ and then use it as the initialization for the optimization objective (Eq. 2). So the predictive model we have trained serves as a fast bottom-up initialization method for a non-convex optimization problem. Figure 3 shows a comparison of these three methods. See Sect. 7.4 for a more quantitative evaluation.

Fig. 3. Projecting real photos onto the image manifold using GAN. Top row: original photos (from handbag dataset); 2nd row: reconstruction using optimization-based method; 3rd row: reconstruction via learned deep encoder P; bottom row: reconstruction using the hybrid method (ours). We show the reconstruction loss below each image.

4.2 Manipulating the Latent Vector

With the image x_0^R projected onto the manifold $\tilde{\mathbb{M}}$ as $x_0 = G(z_0)$ via the projection methods just described, we can start modifying the image on that manifold. We update the initial projection x_0 by simultaneously matching the user intentions while staying on the manifold, close to the original image x_0.

Each editing operation is formulated as a constraint $f_g(x) = v_g$ on a local part of the output image x. The editing operations g include color, shape and warping constraints, and are further described in Sect. 5.1. Given an initial projection x_0, we find a new image $x \in \mathbb{M}$ close to x_0 trying to satisfy as many constraints as possible

$$x^* = \arg\min_{x \in \mathbb{M}} \Big\{ \underbrace{\sum_g \|f_g(x) - v_g\|^2}_{\text{data term}} + \underbrace{\lambda_s \cdot S(x, x_0)}_{\substack{\text{manifold} \\ \text{smoothness}}} \Big\}, \tag{4}$$

where the data term measures deviation from the constraint and the smoothness term enforces moving in small steps on the manifold, so that the image content is not altered too much. We set $\lambda_s = 5$ in our experiments.

The above equation simplifies to the following on the approximate GAN manifold $\tilde{\mathbb{M}}$:

$$z^* = \arg\min_{z \in \mathbb{Z}} \Big\{ \underbrace{\sum_g \|f_g(G(z)) - v_g\|^2}_{\text{data term}} + \underbrace{\lambda_s \cdot \|z - z_0\|^2}_{\substack{\text{manifold} \\ \text{smoothness}}} + E_D \Big\}. \tag{5}$$

Here the last term $E_D = \lambda_D \cdot \log(1 - D(G(z)))$ optionally captures the visual realism of the generated output as judged by the GAN discriminator D.

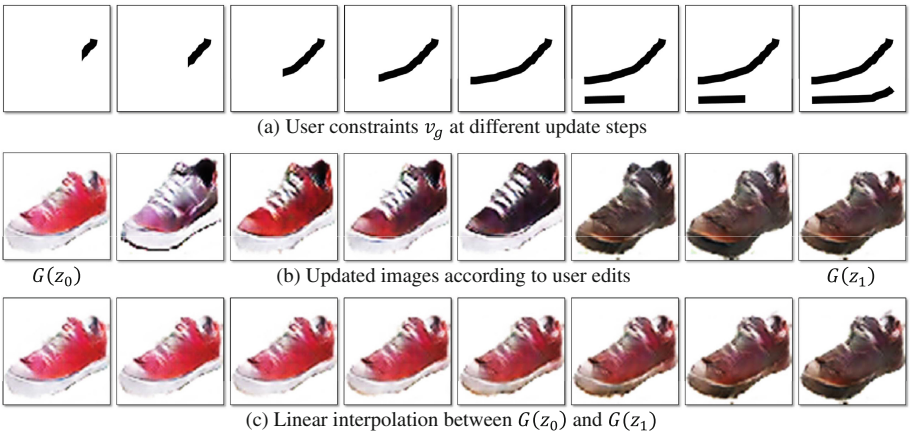

Fig. 4. Updating latent vector given user edits. (a) Evolving user constraint v_g (black color strokes) at each update step; (b) intermediate results at each update step ($G(z_0)$ at leftmost, and $G(z_1)$ at rightmost); (c) a smooth linear interpolation in latent space between $G(z_0)$ and $G(z_1)$.

This further pushes the image towards the manifold of natural images, and slightly improves the visual quality of the result. By default, we turn off this term to increase frame rates.

Gradient descent update: For most constraints Eq. 5 is non-convex. We solve it using gradient descent, which allows us to provide the user with a real-time feedback as she manipulates the image. As a result, the objective 5 evolves in real-time as well. For computational reasons, we only perform a few gradient descent updates after changing the constraints v_g. Each update step takes 50–100 ms, which ensures an interactive feedback. Figure 4 shows one example of the update of z. Given an initial red shoe as shown in Fig. 4, the user gradually scribbles a black color stroke (i.e. specifies a region is black) on the shoe image (Fig. 4a). Then our update method smoothly changes the image appearance (Fig. 4b) by adding more and more of the user constraints. Once the final result $G(z_1)$ is computed, a user can see the interpolation sequence between the initial point z_0 and z_1 (Fig. 4c), and select any intermediate result as the new starting point. Please see supplemental video for more details.

While this editing framework allows us to modify any generated image on the approximate natural image manifold $\tilde{\mathbb{M}}$, it does not directly provide us a way to modify the original high resolution image x_0^R. In the next section we show how edits on the approximate manifold can be transferred to the original image.

4.3 Edit Transfer

Give the original photo x_0^R (e.g. a black shoe) and its projection on the manifold $G(z_0)$, and a user modification $G(z_1)$ by our method (e.g. the generated red shoe).

The generated image $G(z_1)$ captures the roughly change we want, albeit the quality is degraded w.r.t the original image.

Can we instead adjust the original photo and produce a more photo-realistic result x_1^R that exhibits the changes in the generated image? A straightforward way is to transfer directly the pixel changes (i.e. $x_1^R = x_0^R + (G(z_1) - G(z_0))$. We have tried this approach and it introduces new artifacts due to the misalignment of the two images. To address this issue, we develop a dense correspondence algorithm to estimate both the geometric and color changes induced by the editing process.

Specifically, given two generated images $G(z_0)$ and $G(z_1)$, we can generate any number of intermediate frames $\left[G((1-\frac{t}{N})\cdot z_0 + \frac{t}{N}\cdot z_1)\right]_{t=0}^{N}$, where consecutive frames only exhibit minor visual variations.

Motion+Color flow algorithm: We then estimate the color and geometric changes by generalizing the brightness constancy assumption in traditional optical flow methods [30,31]. This results in the following motion+color flow objective[1]:

$$\iint \underbrace{\|I(x,y,t) - A\cdot I(x+u, y+v, t+1)\|^2}_{\text{data term}} + \underbrace{\sigma_s(\|\nabla u\|^2 + \|\nabla v\|^2)}_{\text{spatial reg}} + \underbrace{\sigma_c\|\nabla A\|^2}_{\text{color reg}} dxdy, \quad (6)$$

where $I(x, y, t)$ denotes the RGB values $(r, g, b, 1)^T$ of pixel (x, y) in the generated image $G((1 - \frac{t}{N}) \cdot z_0 + \frac{t}{N} \cdot z_1)$. (u, v) is the flow vector with respect to the change of t, and A denotes a 3×4 color affine transformation matrix. The data term relaxes the color constancy assumption by introducing a locally affine color transfer model A [32] while the spatial and color regularization terms encourage smoothness in both the motion and color change. We solve the objective by iteratively estimating the flow (u, v) using a traditional optical flow algorithm, and computing the color change A by solving a system of linear equations [32]. We iterate 3 times. We produce 8 intermediate frames (i.e. $N = 7$).

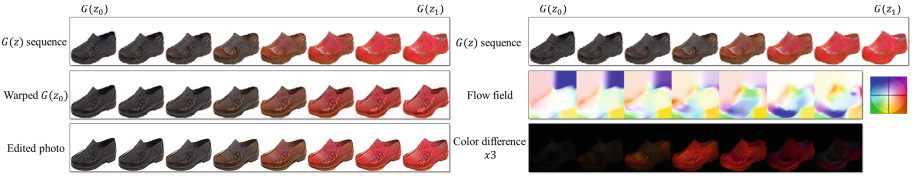

Fig. 5. Edit transfer via Motion+Color Flow. Following user edits on the left shoe $G(z_0)$ we obtain an interpolation sequence in the generated latent space $G(z)$ (top right). We then compute the motion and color flows (right middle and bottom) between neighboring images in $G(z)$. These flows are concatenated and, as a validation, can be applied on $G(z_0)$ to obtain a close reconstruction of $G(z)$ (left middle). The bottom left row shows how the edit is transferred to the original shoe using the same concatenated flow, to obtain a sequence of edited shoes.

[1] For simplicity, we omit the pixel subscript (x, y) for all the variables.

We estimate the changes between nearby frames, and concatenate these changes frame by frame to obtain long-range changes between any two frames along the interpolation sequence $z_0 \rightarrow z_1$. Figure 5 shows a warping sequence after we apply the flow to the initial projection $G(z_0)$.

Transfer edits to the original photo: After estimating the color and shape changes in the generated image sequence, we apply them to the original photo and produce an interesting transition sequence of photo-realistic images as shown in Fig. 5. As the resolution of the flow and color fields are limited to the resolution of the generated image (i.e. 64×64), we upsample those edits using a guided image filter [33].

5 User Interface

The user interface consists of a main window showing the current edited photo, a display showing thumbnails of all the candidate results, and a slider bar to explore the interpolation sequence between the original photo and the final result. Please see our supplemental video for more details.

Candidate results: Given the objective (Eq. 5) derived with the user guidance, we generate multiple different results by initializing z as random perturbations of z_0. We generate 64 examples and show the best 9 results sorted by the objective cost (Eq. 5).

Relative edits: Once a user finishes one edit, she can drag a slider to see all the intermediate results interpolated between the original and the final manipulated photo. We call this "relative edits" as it allows a user to explore more alternatives with a single edit. Similar to relative attributes [34], a user can express ideas like changing the handle of the handbag to be more red, or making the heel of the shoes slightly higher, without committing to a specific final state.

5.1 Editing Constraints

Our system provides three constraints to edit the photo in different aspects: coloring, sketching and warping. All constraints are expressed as brush tools. In the following, we explain the usage of each brush, and the corresponding constraints.

Coloring brush: The coloring brush allows the user to change the color of a specific region. The user selects a color from a palette and can adjust the brush size. For each pixel marked with this brush we constrain the color $f_g(I) = I_p = v_g$ of a pixel p to the selected values v_g.

Sketching brush: The sketching brush allows the user to outline the shape or add fine details. We constrain $f_g(I) = HOG(I)_p$ a differentiable HOG descriptor [35] at a certain location p in the image to be close to the user stroke (i.e. $v_g = HOG(stroke)_p$). We chose the HOG feature extractor because it is binned, which makes it robust to sketching inaccuracies.

Warping brush: The warping brush allows the user to modify the shape more explicitly. The user first selects a local region (a window with adjustable size), and then drag it to another location. We then place both a color and sketching constraint on the displaced pixels encouraging the target patch to mimic the appearance of the dragged region.

Figure 8 shows a few examples where the coloring and sketching brushed were used in the context of interactive image generation. Figure 1 shows the result of the warping brush that was used to pull the topline of the shoe up. Figure 6 shows a few more examples.

6 Implementation Details

Network architecture: We follow the same architecture of deep convolutional generative adversarial networks (DCGAN) [2]. DCGAN mainly builds on multiple convolution, deconvolution and ReLU layers, and eases the min-max training via batch normalization [36]. We train the generator G to produce a $64 \times 64 \times 3$ image given a 100-dimensional random vector. Notice that our method can also use other generative models (e.g. variational auto-encoder [3] or future improvements in this area) to approximate the natural image manifold.

Computational time: We run our system on a Titan X GPU. Each update of the vector z takes $50 \sim 100$ ms, which allows the real-time image editing and generation. Once an edit is finished, it takes $5 \sim 10$ s for our edit transfer method to produce high-resolution final result.

7 Results

We first introduce the statistics of our dataset. We then show three main applications: realistic image manipulation, generative image transformation, and generating a photo from scratch using our brush tools. Finally, we evaluate our image reconstruction methods, and perform a human perception study to understand the realism of generated results. Please refer to the supplementary material for more results and comparisons.

Datasets: We experiment with multiple photo collections from various sources as follows: "shoes" dataset [37], which has 50 K shoes collected from Zappos.com (the shoes are roughly centered but not well aligned, and roughly facing left, with frontal to side view); "church outdoor" dataset (126 K images) from the LSUN challenge [38]; "outdoor natural" images (150 K) from the MIT Places dataset [39]; and two query-based product collections downloaded from Amazon, including "handbags" (138 K) and "shirts" (137 K). The downloaded handbags and shirts are roughly centered but no further alignment has been performed.

7.1 Image Manipulation

Our main application is photo-realistic image manipulation using the brush interactions described in Sect. 5.1. See Fig. 6 for a few examples where the brush edits are depicted on the left (dashed line for the sketch tool, color scribble for the color brush and a red square with arrow for the warp tool). See the supplementary video for more interactive manipulation demos.

Fig. 6. Image manipulation examples: for each example, we show the original photo and user edits on the left. The top row on the right shows the generated sequence and the bottom row shows the edit transfer sequence on the original image. (Color figure online)

7.2 Generative Image Transformation

An interesting outcome of the editing process is the sequence of intermediate generated images that can be seen as a new kind of image morphing [11,12,40]. We call it "generative transformation". We use this sequence to transform the shape and color of one image to look like another image automatically, i.e., *without* any user edits. This is done by applying the motion+color flow on either of the sources. Figure 7 shows a few "generative transform" examples.

7.3 Interactive Image Generation

Another byproduct of our method is that if there is no image to begin with and all we have are the user brush strokes, the method would generate a natural image that best satisfies the user constraints. This could be useful for dataset exploration and browsing. The difference with previous sketch-to-image retrieval methods [41] or AverageExplorer [42], is that due to potentially contradicting

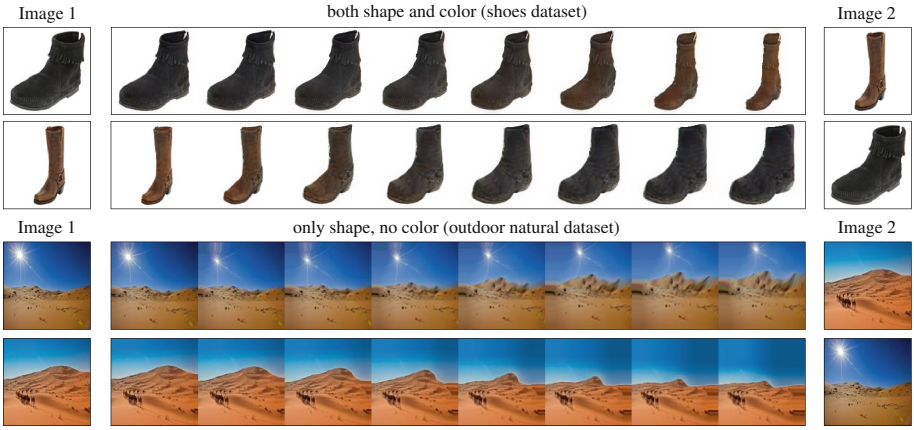

Fig. 7. Generative image transformation. In both rows, the source on the left is transformed to have the shape and color (or just shape in the 2nd example) of the one on the right.

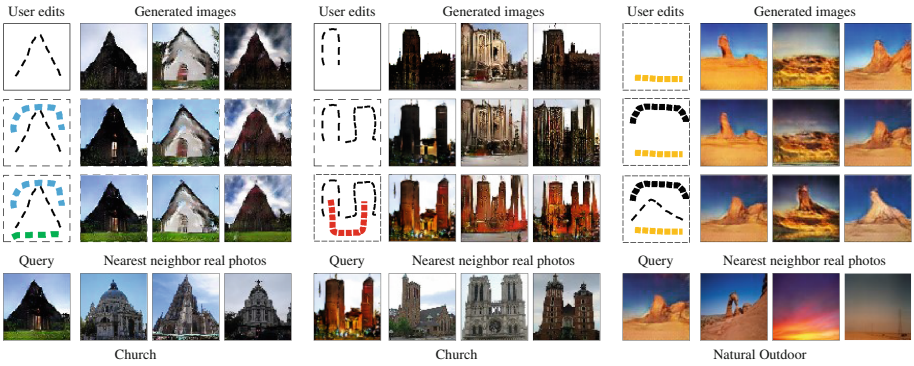

Fig. 8. Interactive image generation. The user uses the brush tools to generate an image from scratch (top row) and then keeps adding more scribbles to refine the result (2nd and 3rd rows). In the last row, we show the most similar real images to the generated images. (dashed line for the sketch tool, and color scribble for the color brush)

user constraints, the result may look very different than any single image from the dataset or an average of such images, and more of a realistic hybrid image [43]. See some examples in Fig. 8.

7.4 Evaluation

Image reconstruction evaluation: We evaluate three image reconstruction methods described in Sect. 4.1: optimization-based, network-based and our hybrid approach that combines the last two. We run these on 500 test images per

Table 1. Average per-dataset image reconstruction error measured by $\mathcal{L}(x, x^R)$.

	Shoes	Church outdoor	Outdoor natural	Handbags	Shirts
Optimization-based	0.155	0.319	0.176	0.299	0.284
Network-based	0.210	0.338	0.198	0.302	0.265
Hybrid (ours)	**0.140**	**0.250**	**0.145**	**0.242**	**0.184**

category, and evaluate them by the reconstruction error $\mathcal{L}(x, x^R)$ defined in Eq. 1. Table 1 shows the mean reconstruction error of these three methods on 5 different datasets. We can see the optimization-based and neural network-based methods perform comparably, where their combination yields better results. See Figure 3 for a qualitative comparison. We include PSNR (in dB) results in the supplementary material.

Class-specific model: So far, we have trained the generative model on a particular class of images. As a comparison, we train a cross-class model on three datasets altogether (i.e. shoes, handbags, and shirts), and observe that the model achieves worse reconstruction error compared to class-specific models (by $\sim 10\,\%$). We also have tried to use a class-specific model to reconstruct images from a different class. The mean cross-category reconstruction errors are much worse: shoes model used for shoes: 0.140 vs. shoes model for handbags: 0.398, and for shirts: 0.451. However, we expect a model trained on many categories (e.g. $1,000$) to generalize better to novel objects.

Perception study: We perform a small perception study to compare the photo realism of four types of images: real photos, generated samples produced by GAN, our method (shape only), and our method (shape+color). We collect 20 annotations for 400 images by asking Amazon Mechanical Turk workers if the image look realistic or not. Real photos: 91.5 %, DCGAN: 14.3 %, ours (shape+color): 25.9 %; ours (shape only): 48.7 %. DCGAN model alone produces less photo-realistic images, but when combined with our edit transfer, the realism significantly improves.

Additional evaluation: In the supplemental material, we evaluate our motion+color flow method, and compare our results against popular alignment methods that are designed to handle large displacement between two images [44, 45].

8 Discussion and Limitations

We presented a step towards image editing with a direct constraint to stay close to the manifold of real images. We approximate this manifold using the state-of-the-art in deep generative models (DCGAN). We show how to make interactive edits to the generated images and transfer the resulting changes in shape and color back to the original image. Thus, the quality of the generated results

(low resolution, missing texture and details) and the types of data DCGAN is applicable to (works well on structured datasets such as product images and worse on more general imagery), limits how far we can get with this editing approach. However our method is not tied to a particular generative method and will improve with the advancement of this field. Our current editing brush tools allow rough changes in color and shape but not texture and more complex structure changes. We leave these for future work.

Acknowledgments. This work was supported, in part, by funding from Adobe, eBay and Intel, as well as a hardware grant from NVIDIA. J.-Y. Zhu is supported by Facebook Graduate Fellowship.

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS. 2672–2680. (2014)
2. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR (2016)
3. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
4. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models usinga laplacian pyramid of adversarial networks. In: NIPS, pp. 1486–1494 (2015)
5. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. arXiv preprint arXiv:1602.02644 (2016)
6. Reinhard, E., Ashikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Comput. Graph. Appl. **21**, 34–41 (2001)
7. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: SIGGRAPH, SIGGRAPH 2004, pp. 689–694. ACM, New York (2004)
8. Alexa, M., Cohen-Or, D., Levin, D.: As-rigid-as-possible shape interpolation. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000 (2000)
9. Krähenbühl, P., Lang, M., Hornung, A., Gross, M.: A system for retargeting of streaming video. In: ACM Trans. Graph. (TOG), vol. 28. p. 126. ACM (2009)
10. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: Patchmatch: a randomized correspondence algorithm for structural image editing. SIGGRAPH **28**(3), 24 (2009)
11. Wolberg, G.: Digital Image Warping. IEEE Computer Society Press, Los Alamitos (1990)
12. Shechtman, E., Rav-Acha, A., Irani, M., Seitz, S.: Regenerative morphing. In: CVPR, San-Francisco, CA, June 2010
13. Kemelmacher-Shlizerman, I., Shechtman, E., Garg, R., Seitz, S.M.: Exploring photobios. In: SIGGRAPH, vol. 30, p. 61 (2011)
14. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**, 607–609 (1996)
15. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. IJCV **40**(1), 49–70 (2000)
16. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: Proceedings of ICCV, pp. 479–486 (2011)
17. Roth, S., Black, M.J.: Fields of experts: a framework for learning image priors. In: CVPR (2005)

18. Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Learning a discriminative model for the perception of realism in composite images. In: ICCV (2015)
19. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
20. Salakhutdinov, R., Hinton, G.E.: Deep boltzmann machines. In: AISTATS (2009)
21. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: ICML (2008)
22. Bengio, Y., Laufer, E., Alain, G., Yosinski, J.: Deep generative stochastic networks trainable by backprop. In: ICML, pp. 226–234 (2014)
23. Gregor, K., Danihelka, I., Graves, A., Wierstra, D.: Draw: a recurrent neural network for image generation. In: ICML (2015)
24. Dosovitskiy, A., Tobias Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: CVPR, pp. 1538–1546 (2015)
25. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. arXiv preprint arXiv:1603.08155 (2016)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS, pp. 1097–1105 (2012)
27. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: CVPR, pp. 248–255. IEEE (2009)
28. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**(5), 1190–1208 (1995)
29. Gershman, S.J., Goodman, N.D.: Amortized inference in probabilistic reasoning. In: Proceedings of the 36th Annual Conference of the Cognitive Science Society (2014)
30. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: Pajdla, T., Matas, J.G. (eds.) ECCV 2004. LNCS, vol. 3024, pp. 25–36. Springer, Heidelberg (2004)
31. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/kanade meets horn/schunck: combining local and global optic flow methods. IJCV **61**(3), 211–231 (2005)
32. Shih, Y., Paris, S., Durand, F., Freeman, W.T.: Data-driven hallucination of different times of day from a single outdoor photo. ACM Trans. Graph. (TOG) **32**(6), 200 (2013)
33. He, K., Sun, J., Tang, X.: Guided image filtering. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 1–14. Springer, Heidelberg (2010)
34. Parikh, D., Grauman, K.: Relative attributes. In: ICCV, pp. 503–510. IEEE (2011)
35. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893. IEEE (2005)
36. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. ICML **37**, 448–456 (2015)
37. Yu, A., Grauman, K.: Fine-grained visual comparisons with local learning. In: CVPR, pp. 192–199 (2014)
38. Yu, F., Zhang, Y., Song, S., Seff, A., Xiao, J.: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015)
39. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: NIPS, pp. 487–495 (2014)
40. Seitz, S.M., Dyer, C.R.: View Morphing, pp. 21–30, New York (1996)
41. Sun, X., Wang, C., Xu, C., Zhang, L.: Indexing billions of images for sketch-based retrieval. In: ACM MM (2013)

42. Zhu, J.Y., Lee, Y.J., Efros, A.A.: Averageexplorer: interactive exploration and alignment of visual data collections. SIGGRAPH **33(**4) (2014)
43. Risser, E., Han, C., Dahyot, R., Grinspun, E.: Synthesizing structured image hybrids. SIGGRAPH **29**(4), 85:1–85:6 (2010)
44. Liu, C., Yuen, J., Torralba, A.: Sift flow: dense correspondence across scenes and its applications. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 978–994 (2011)
45. Kim, J., Liu, C., Sha, F., Grauman, K.: Deformable spatial pyramid matching for fast dense correspondences. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2307–2314 (2013)