CG2Real: Improving the Realism of Computer Generated Images Using a Large Collection of Photographs

Micah K. Johnson, *Member*, *IEEE*, Kevin Dale, *Student Member*, *IEEE*, Shai Avidan, *Member*, *IEEE*, Hanspeter Pfister, *Senior Member*, *IEEE*, William T. Freeman, *Fellow*, *IEEE*, and Wojciech Matusik

Abstract—Computer-generated (CG) images have achieved high levels of realism. This realism, however, comes at the cost of long and expensive manual modeling, and often humans can still distinguish between CG and real images. We introduce a new data-driven approach for rendering realistic imagery that uses a large collection of photographs gathered from online repositories. Given a CG image, we retrieve a small number of real images with similar global structure. We identify corresponding regions between the CG and real images using a mean-shift cosegmentation algorithm. The user can then automatically transfer color, tone, and texture from matching regions to the CG image. Our system only uses image processing operations and does not require a 3D model of the scene, making it fast and easy to integrate into digital content creation workflows. Results of a user study show that our hybrid images appear more realistic than the originals.

Index Terms—Image enhancement, image databases, image-based rendering.

1 Introduction

The field of image synthesis has matured to the point where photo-realistic computer-generated (CG) images can be produced with commercially available software packages (e.g., RenderMan and POV-Ray). However, reproducing the details and quality of a natural image requires a considerable amount of time by a skilled artist. Even with large budgets and many man-hours of work, it is sometimes surprisingly easy to distinguish CG images from photographs.

CG images differ from photographs in three major ways. First, color distributions of CG images are often overly saturated and exaggerated. Second, multiscale image statistics, such as histograms of filter outputs at different scales, rarely match the statistics of photographs. Finally, CG images often lack details (e.g., high frequencies, texture, and noise) and consequently look too pristine.

To improve realism in computer graphics, rather than introduce more detailed geometric models and complex textures, this work proposes a new rendering pipeline that leverages a large collection of photographs. Large image collections can be readily acquired from photo-sharing sites, such as Flickr, and such collections have been the basis for data-driven methods for improving photographs [1]. This work demonstrates that large image collections can be used in the context of computer graphics to synthesize realistic imagery without the need for complex models.

CG2Real takes a CG image, retrieves and aligns a small number of similar photographs from a database, and transfers the color, tone, and texture from the real images to the CG image. This process can improve the realism of a CG image, as shown in Fig. 1. A key ingredient in the system is a mean-shift cosegmentation algorithm that matches regions in the CG image with regions in the real images. After cosegmentation, we transfer real-image textures to the CG image and perform local color and tone transfers between image regions. Local transfers offer an improvement in quality over global transfers based on histogram matching. Color and tone transfers are completely automatic, and texture transfer can be controlled by adjusting a few parameters. In addition, all operations are reasonably fast: an average computer can run the cosegmentation and all three transfer operations in less than 15 seconds for a 600 × 400 pixel image.

CG2Real is useful in a variety of artistic scenarios. In its current form, the system can synthesize natural scenes using a database of outdoor images. The CG image is used as a template that gets filled in with realistic textures. The user can also choose to preserve structures in the CG image. For example, a user working on a 3D model may wish to render it with a photo-realistic background. This problem occurs in architectural modeling where an architect has a detailed model for a house but not for the surroundings. In this scenario, CG2Real can be configured to preserve the rendered model and synthesize a background using real image textures (e.g., grass, trees, and sky). CG2Real allows a

- *M.K. Johnson, W.T. Freeman, and W. Matusik are with the Massachusetts Institute of Technology, 77 Massachusetts Ave, MIT, Cambridge, MA 02139.*
E-mail: kimo@csail.mit.edu, billf@mit.edu, matusik@disneyresearch.com.
- *K. Dale and H. Pfister are with Harvard University, 33 Oxford Street, Cambridge, MA 02138.*
E-mail: dale@eecs.harvard.edu, pfister@seas.harvard.edu.
- *S. Avidan is with the Department of Electrical Engineering, Faculty of Engineering, Tel Aviv University, Tel Aviv 69978, Israel, and Adobe Systems. E-mail: shai.avidan@gmail.com.*

Manuscript received 8 Mar. 2010; revised 13 June 2010; accepted 10 July 2010; published online 26 Oct. 2010.
Recommended for acceptance by M. Chen.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-03-0061.
Digital Object Identifier no. 10.1109/TVCG.2010.233.

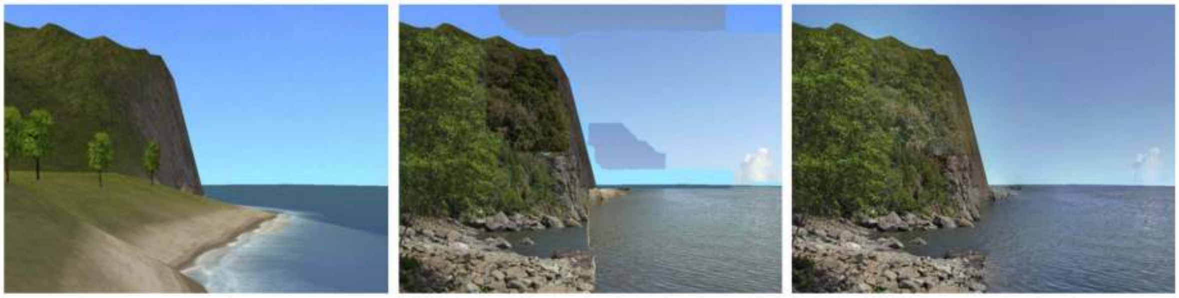


Fig. 1. Given an input CG image (left), our system finds the most similar photographs (not shown) to the input image. Next, the system identifies similar regions between the CG image and photographs, transfers these regions into the CG image (center), and uses seamless compositing to blend the regions. Finally, it transfers local color and gradient statistics from the photographs to the input image to create a color and tone adjusted image (right).

user to control the amount of image content to be replaced by real textures, enabling artists to create hybrid images and enabling researchers to study the cues important to the perception of realism.

The primary contribution of this paper is a novel data-driven approach for rendering realistic imagery based on a CG input. Within this system, several individual operations also further the state of the art, including 1) an improved image search tuned for matching global image structure between CG and real images; 2) an image cosegmentation algorithm that is both fast and sufficiently accurate for color and tone transfer; and 3) methods for local transfer of color, tone, and texture that take advantage of region correspondences. As a final contribution, we describe several user studies that demonstrate that our hybrid images appear more realistic than the originals, providing insight into the cues that people use to distinguish CG from real.

2 Previous Work

The appearance of a computer-generated image can be improved by adding realistic texture. In their seminal work, Heeger and Bergen [2] proposed a novel approach for synthesizing textures. Their method starts with a random noise image and iteratively adjust its statistics at different scales to match those of the target texture, leading to new instances of the target texture. This approach was later extended by De Bonet [3] to use joint multiscale statistics. Alternatively, one can take an exemplar-based approach to texture synthesis. This idea was first illustrated in the work of Efros and Leung [4] and was later extended to work on patches instead of pixels [5], [6]. The image analogies framework [7] extends nonparametric texture synthesis by learning a mapping between a given exemplar pair of images and applying the mapping to novel images. Freeman et al. [8] proposed a learning-based approach to solve a range of low-level image processing problems (e.g., image superresolution) that relies on having a dictionary of corresponding patches that is used to process a given image.

Unfortunately, these approaches require correspondence between the source and target images (or patches), a fairly strong assumption that cannot always be satisfied. Rosales et al. [9] relaxed this assumption by framing the problem as a large inference problem, where both the position and appearance of the patches are inferred from a pair of images without correspondence. While the results look convincing for a variety of applications, the specific problem of improving realism in CG images was not addressed.

Instead of requiring corresponding images (or patches) to learn a mapping, one can take a global approach that transfers color or style between images. Reinhard et al. [10] modified the color distribution of an image to give it the appearance of another image. They showed results on both photographic and synthetic images. Alternatively, Pitié et al. [11] posed color transfer as a problem of estimating a continuous N-dimensional transfer function between two probability distribution functions and presented an iterative nonlinear algorithm. Bae et al. [12] used a two-scale nonlinear decomposition of an image to transfer style between images. In their approach, histograms of each layer were modified independently and then recombined to obtain the final output image. Finally, Wen et al. [13] demonstrated a stroke-based interface for performing local color transfer between images. In this system, the user provides the target image and additional input in the form of stroke pairs.

We build on and extend this line of work with several important distinctions. First, the works discussed so far do not consider how the model images are chosen, and instead assume that the user provides them. We believe a system should be able to obtain model images with a minimum of user assistance. Given a large collection of photographs, we find images with similar global structure (e.g., trees next to mountains below a blue sky). Because the photographs are semantically and contextually similar to the CG image, we can find corresponding regions using cosegmentation, and thus can more easily apply local style transfer methods to improve realism.

Recently, several authors have demonstrated the use of large collections of images for image editing operations. The system of Hays and Efros [1] uses a large collection of images to complete missing information in a target image. Their system retrieves a number of images that are similar to the query image and uses them to complete a user-specified region. We take a different approach by automatically identifying matching regions and by stitching together regions from multiple images. Liu et al. [14] perform example-based image colorization using images from the web that is robust to illumination differences.

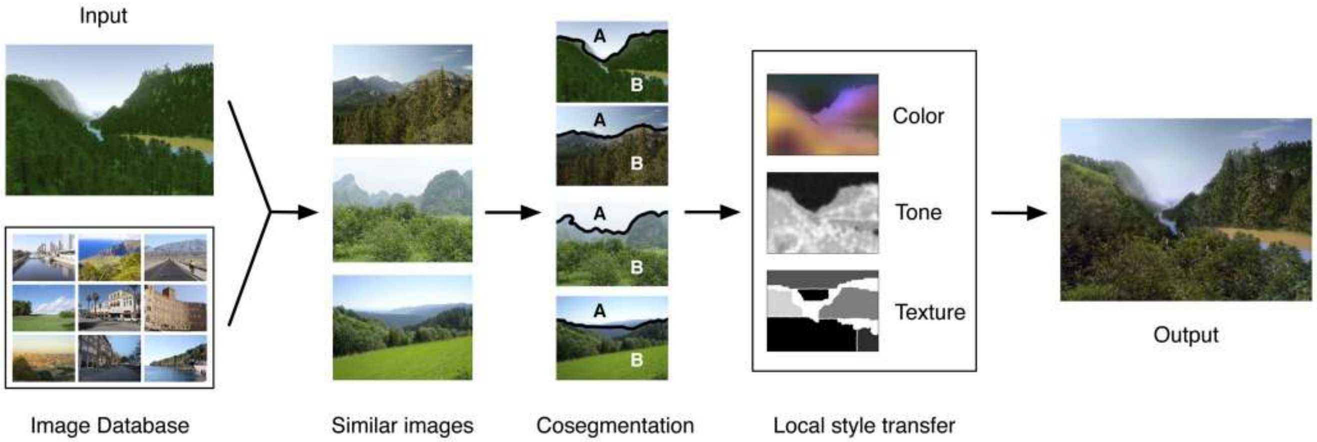


Fig. 2. An overview of our system. We start by querying a large collection of photographs to retrieve the most similar images. The user selects the k closest matches and the images, both real and CG, are cosegmented to identify similar regions. Finally, local style transfer algorithms use the real images to upgrade the color, tone, and/or texture of the CG image.

However, their method involves image registration between search results and input and requires exact scene matches. Our approach instead uses an image search based on image data, and our transfers only assume similar content between CG input and real search results. Finally, Dale et al. [15] show that large image collections can be used to find context-specific priors for images and demonstrate the utility of these priors for global corrections to exposure, contrast, and white balance. In this work, we consider CG images as input and address local transfers of color, tone, and texture.

In work based on annotated image data sets, Lalonde et al. [16] use image regions drawn from the LabelMe database [17] to populate the query image with new objects. Johnson et al. [18] allow the user to create novel composite images by typing in a few nouns at different image locations. A similar system called Sketch2Photo [19] takes sketches with text labels and synthesizes an image from the sketch. These systems rely on image annotations to identify image regions and region correspondences. They can produce decent results when replacing well-defined objects, but are more difficult to use for replacing image regions that cannot be defined by a simple search term. In contrast, our approach uses an image-based search descriptor with an automatic cosegmentation algorithm for identifying local regions and interregion correspondences.

Researchers have also studied the characteristics of natural versus synthetic images. For example, in digital forensics, Lyu and Farid [20] examine high-order image statistics to distinguish between synthetic and natural images. Lalonde and Efros [21] use color information to predict if a composite image will look natural or not. Others have focused solely on learning a model for the statistics of natural images [22], [23]. These works suggest that natural images have relatively consistent statistical properties and that these properties can be used to distinguish between synthetic and natural images. Based on this observation, our color and tone transfer algorithms work statistically, adjusting color and gradient distributions to match corresponding distributions from real images.

3 Image and Region Matching

Fig. 2 shows an overview of our system. First, we retrieve the N closest real images to the query CG image. The N images are shown to the user, who selects the k most relevant images; typically, $N = 30$ and $k = 5$. Next, we perform a cosegmentation of the k real images with the CG image to identify similar image regions. Once the images are segmented, the user chooses among three different types of transfer from the real images to the CG image: texture, color, and tone. We find that all three types of style transfer can improve the realism of low-quality CG images.

3.1 Image Database

Our system leverages a database of 4.5 million natural images crawled from the photo-sharing site Flickr using keywords related to outdoor scenes, such as "beach," "forest," "city," etc. Each image, originally of Flickr's large size with a maximum dimension of 1,024 pixels, was downsampled to approximately 75 percent its original size and stored in PNG format (24-bit color) to minimize the impact of JPEG compression artifacts on our algorithms.

Similar to other work that uses large image collections [1], [24], we chose to focus on a specific scene class; in our case, outdoor scenes. We have found that the performance of our system improves with larger database sizes, and while it is straightforward to build a large database for a targeted image class, it is still computationally impractical to build a large database that densely samples the space of imagery. While our techniques could work on other scene types (e.g., indoor), the variability would require a much larger database to yield usable matching images [15].

To search the database for matching images, we need an efficient, yet descriptive, image representation. The gist scene descriptor [25] is one choice of representation that has been used successfully for image matching tasks [1]. The gist descriptor uses histograms of Gabor filter responses at a single level. We used gist in an early implementation of our system and were not fully satisfied with the results. In a recent study, Gabor-based descriptors, such as gist, were outperformed by SIFT-based descriptors for texture classification [26], justifying our decision to use a more detailed image representation.

Our representation is based on visual words, or quantized SIFT features [27], and the spatial pyramid matching scheme of Lazebnik et al. [28]. This approach has been shown to perform well for semantic scene classification and for finding context-specific priors for real images [15]. To build a descriptor robust to differences between CG and real images, we use a descriptor with significant spatial resolution that favors global structural alignment and we use small visual word vocabularies to coarsely quantize appearance.

Specifically, we use two vocabularies of 10 and 50 words and grid resolutions of 1×1, for the 10-word vocabulary, and $1 \times 1, 2 \times 2, 4 \times 4$, and 8×8, for the 50-word vocabulary, for a final pyramid descriptor with 4,260 elements. This representation has some redundancy, since a visual word will occur multiple times across pyramid levels. The weighting scheme specified by the pyramid match kernel [29] accounts for this; it also effectively provides term-frequency (tf) weighting. We also apply inverse document frequency (idf) weighting to the pyramid descriptor.

In addition, we represent a rough spatial layout of color with an 8×8 downsampled version of the image in CIE L*a*b* space (192 elements). Since the search is part of an interactive system, we use principal component analysis (PCA) to reduce the descriptor dimensionality to allow for an efficient in-core search. We keep 700 elements for the pyramid term and 48 for the color term and L2-normalize each. The final descriptor is the concatenation of the spatial pyramid and color terms, weighted by α and $(1-\alpha)$, respectively, for $\alpha \in [0,1]$. Similarity between two images is measured by Euclidean distance between their descriptors.

For low-quality CG images, texture is only a weak cue, so smaller α values achieve a better balance of color versus texture cues. We found that presenting the user with 15 results obtained with $\alpha = 0.25$ and 15 with $\alpha = 0.75$ yielded a good balance between the quality of matches, robustness to differences in fidelity of CG inputs, and time spent by the user during selection. We use a kd tree-based exact nearest-neighbor search, which requires about 2 seconds per query on a 3 GHz dual-core machine.

While more complex than gist [25], we find that our descriptor consistently returns better images for the same query image. To quantify the improvement in search results, we conducted a user study where the users were asked to judge the top 20 search results from each algorithm. The full details of this study are described in Section 6.2.

3.2 Cosegmentation

Global transfer operations between two images, such as color and tone transfer, work best when the images have similarly sized regions, e.g., when there are similar amounts of sky, ground, or buildings. If the images have different regions, or if one image contains a large region that is not in the other image, global transfers can fail. Similar to Tai et al. [30], we find that segmenting the images and identifying regional correspondences before color transfer greatly improves the quality and robustness of the results. But in contrast to their work, we use cosegmentation to segment and match regions in a single step. This approach is better than segmenting each image independently and matching regions after the fact because the content of all images is taken into account during cosegmentation and matching regions are automatically produced as a byproduct [31].

The cosegmentation approach of Rother et al. [31] uses an NP-hard energy function with terms to encode both spatial coherency and appearance histograms. To optimize it, they present a novel scheme that they call trust-region graph cuts. It uses an approximate minimization technique to obtain an initial estimate and then refines the estimate in the dual space to the original problem.

While our goal is similar to that of Rother et al., we use the simple approach of Dale et al. [15] that is based on the mean-shift framework [32]. We define a feature vector at every pixel p that is the concatenation of the pixel color in L*a*b* space, the normalized x and y coordinates at p, and a binary indicator vector $(i_0, \ldots, i_k)$ such that i_j is 1 when pixel p is in the jth image and 0 otherwise. Note that the problem of segmenting a set of related images is different from the problem of segmenting video—there is no notion of distance across the image index dimension as there is in a video stream (i.e., there is no time dimension). Thus, the final components of the feature vector only differentiate between pixels that come from the same image versus those that come from different images.

The distance between feature vectors at pixels p_1 and p_2 in images I_j and I_k is a weighted Euclidean distance:

$$\begin{aligned} d(p_1, I_j, p_2, I_k)^2 = {} & \sigma_c^2 d_c(I_j(p_1), I_k(p_2))^2 \\ & + \sigma_s^2 d_s(p_1, p_2)^2 + \sigma_b^2 \delta(j-k), \end{aligned} \tag{1}$$

where $d_c(I_j(p_1), I_k(p_2))$ is the L*a*b* color distance between pixel p_1 in image I_j and pixel p_2 in image I_k and $d_s(p_1, p_2)$ is the spatial distance between pixels p_1 and p_2. The delta function encodes the distance between the binary components of the feature vector.

The scalars σ_c, σ_s, and σ_b in (1) serve as weights to balance the color, spatial, and binary index components. The default values are $\sigma_c = 0.4$, $\sigma_s = 0.5$, and $\sigma_b = 0.1$, and we find that the weights and the mean-shift bandwidth parameter do not need to be adjusted for individual image sets to achieve the types of segmentations that are useful to our color and tone transfer algorithms.

A disadvantage of mean-shift is that it can be costly to compute at every pixel of an image without using specific assumptions about feature vectors or kernel [33]. Since we are after coarse regional correspondences, we reduce the size of the image by a factor of 8 along each dimension and use a standard mean-shift algorithm with the feature vectors described above. We then upsample the cosegmentation maps to full resolution using joint bilateral upsampling [34].

In Fig. 3, we show three cosegmentation results, each with three images (one CG, two real). In the first two cases, the algorithm segments the images into sky and nonsky. In the last case, the images are segmented into three regions: ground, sky, and water. In all cases, the segment correspondences are correct, and although our color and tone transfer algorithms are robust to it, the images have not been oversegmented.

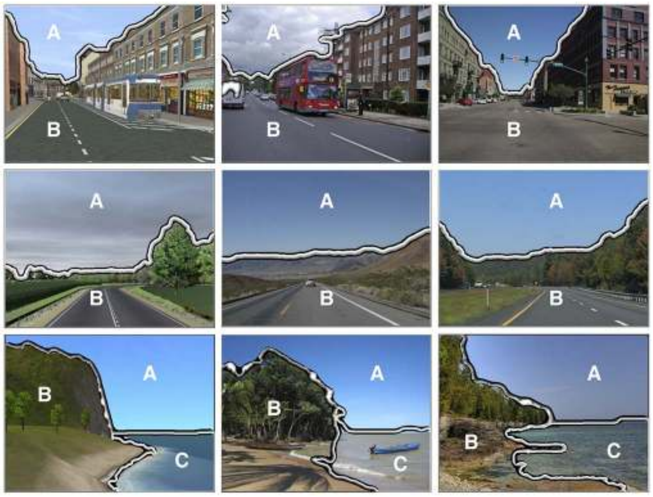


Fig. 3. Results from our cosegmentation algorithm. In each row, the CG image is shown on the left and two real image matches, on the right. Note that in all cases, segment correspondences are correct, and the images are not oversegmented.

4 Local Style Transfer Operators

After cosegmentation, we apply local style transfer operations for color, tone, and texture.

4.1 Local Color Transfer

The simplest style transfer is color transfer, where colors of the real images are transferred to the CG image by transferring the statistics of a multidimensional histogram. This method was shown to work quite well for color transfer between *real* images, but it often fails when applied to CG images. The main difficulty is that the color histogram of CG images is typically different from the histogram of real images—it is much more sparse (fewer colors are used). The sparsity and simplicity of the color distributions can lead to instability during global transfer where colors are mapped arbitrarily, as shown in the bottom row of Fig. 4.

We mitigate these problems by a combination of joint-bilateral upsampling and local color transfer. We downsample the images, compute the color transfer offsets per region from the lower resolution images, and then smooth and upsample the offsets using joint bilateral upsampling. Working on regions addresses the problem of images that contain different proportions of colors and joint bilateral upsampling smooths color transfer in the spatial domain.

Within each subsampled region, our color transfer algorithm uses 2D histogram matching on the a^* and b^* channels, and 1D histogram matching on the L^* channel. The advantage of histogram matching methods is that they do not require per pixel correspondences, which we do not have. Unfortunately, unlike 1D histogram matching, there is no closed form solution for 2D histogram transfer. We use an iterative algorithm by Pitié et al. [11] that projects the 2D histogram onto random 1D axes, performs standard 1D histogram matching, and reprojects the data back to 2D. The algorithm typically converges in fewer than 10 iterations. We found that marginalizing the distributions and performing the remapping independently for the a^* and b^* channels produces inferior results.

In Fig. 4, we show two examples of color transfer. From left to right, we show the original CG images, the real images used as color models, the results of global N-dimensional color transfer (in $L^*a^*b^*$ space), and results of our region-based transfer. In the top example, the global algorithm produces a blue color cast over the entire image because the real image has significantly more sky than the CG image. In the bottom example, the house and sky are influenced by the opposite regions in the model image: the house becomes blue and the sky becomes a neutral gray. These problems are avoided by local color transfer.

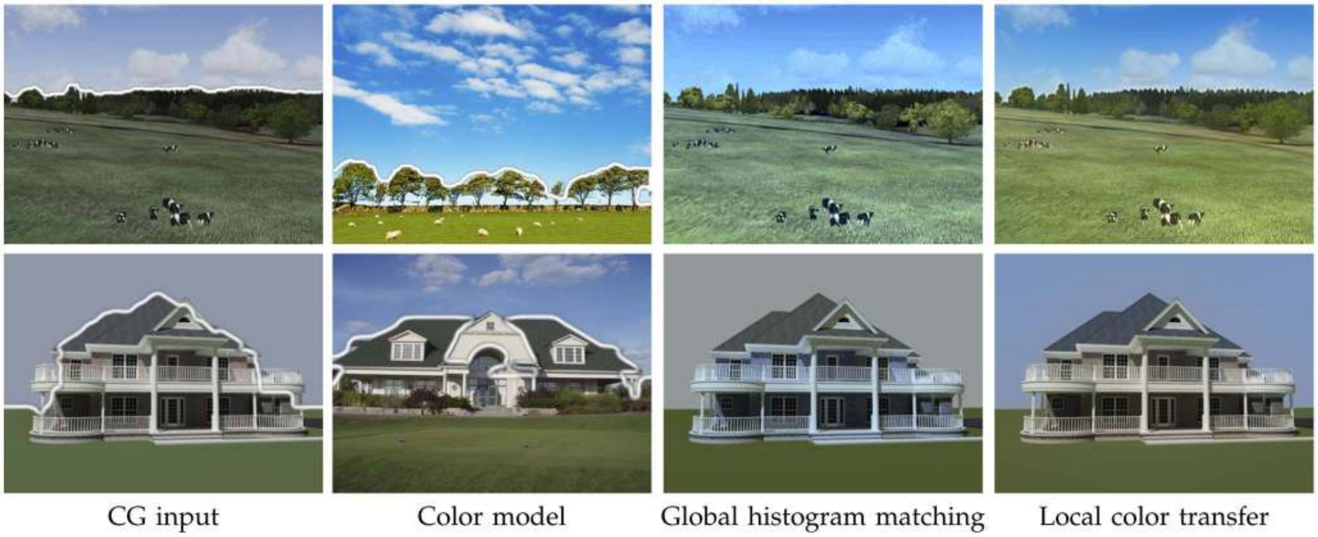


Fig. 4. Transferring image color using cosegmentation. On the left are CG images and real images that serve as color models; white lines are superimposed on the images to denote the cosegmentation boundaries. On the right are results from two color transfer algorithms: a global algorithm based on N-dimensional histogram matching [11], and our local color transfer algorithm. In the top example, the global result has a bluish color cast. In the bottom example, the global result swaps the colors of the building and the sky. Local color transfer yields better results on both examples.

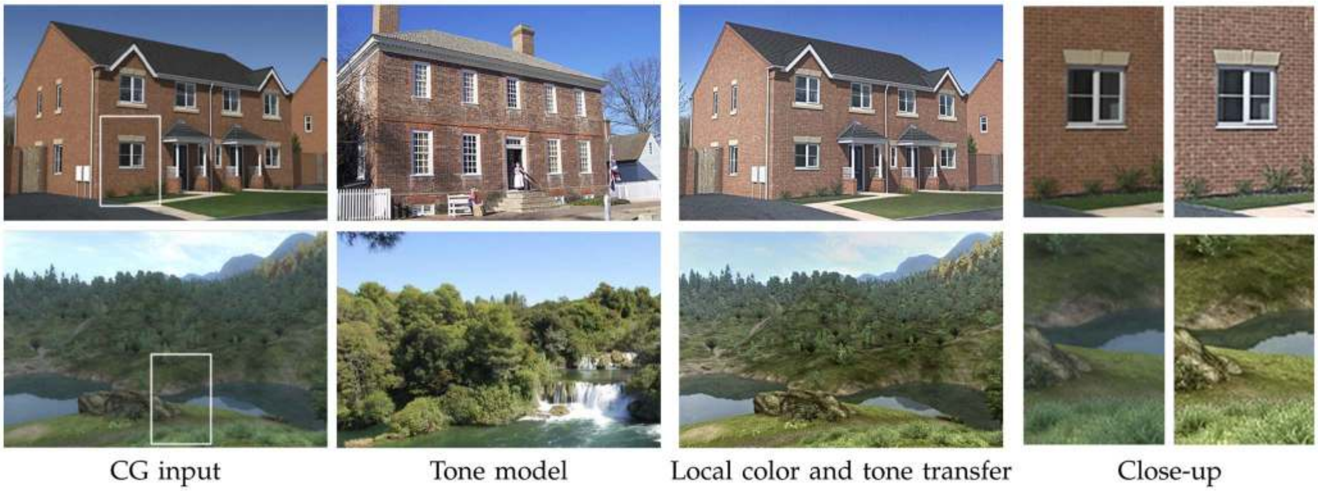


Fig. 5. Transferring image tone. From left to right: an input CG image, a real image that will serve as a color and tone model, the result of region-based transfer of subband distributions, and close-up view of before and after tone transfer.

4.2 Local Tone Transfer

In addition to transferring the color histogram from the photographs to the CG image, we are also interested in adjusting gradient histograms at different scales. To this end, we apply a method similar to Bae et al. [12] to match filter statistics of the luminance channel of the CG image to the photographs. Our method, however, transfers detail locally within cosegmentation regions and uses a four-level pyramid based on a quadrature mirror filter [35] instead of a bilateral filter.

Our tone transfer algorithm is similar to the algorithm for color transfer. First, we decompose the luminance channel of the CG image and one or more real images using a QMF pyramid. Next, we use 1D histogram matching to match the subband statistics of the CG image to the real images in every region. After transferring on subbands, we model the effect of histogram transfer on subband signals as a change in gain:

$$s'_i(p) = g_i(p)s_i(p), \tag{2}$$

where $s_i(p)$ is the level i subband coefficient at pixel p, and $s'_i(p)$ is the corresponding subband coefficient after regional histogram matching, and $g_i(p)$ is the gain. Gain values greater than one will amplify detail in that subband and gain values less than one will diminish detail. To avoid halos or other artifacts, we employ the gain-scaling strategies described by Li et al. [36] to ensure that lower subbands are not amplified beyond higher subbands and that the gain signals are smooth near zero-crossings.

The results of color and tone transfer are shown in Fig. 5. As can be seen, the appearance of the CG image changes subtly. Since color and tone transfers do not fundamentally change the structure of the image, they can be used even when the image matches returned from the database are poor or when the CG image is already close to being photo-realistic.

4.3 Local Texture Transfer

In addition to color and tone transfer, we also transfer texture from photographs, which improves realism especially for low-quality CG images. This is different from texture synthesis, where the goal is to synthesize more of the same texture given an example texture. In our case, we do not want to reuse the same region many times because this often leads to visual artifacts in the form of repeated regions. Instead, we rely on the k similar photographs we retrieved from the database to provide us with a set of textures to help upgrade the realism of the CG image.

We start local texture transfer by aligning the CG image with the real images. Transferring texture on a region-by-region basis, as was done for color and tone, does not work well because region boundaries do not always correspond to strong edges in the image. As a result, slightly different textures can be transferred to neighboring regions, leading to noticeable visual artifacts. To reduce these artifacts, we align multiple shifted copies of each real image to the different regions of the CG image and transfer textures using graph cut. The result is a coherent texture transfer that respects strong scene structure.

We perform the region-based alignment as follows: For each cosegmented region in the CG image, we use cross-correlation of edge maps (magnitudes of gradients) to find the real image, and the optimal shift, that best matches the CG image for that particular region. We repeat the process in a greedy manner until all regions in the CG image are completely covered. To reduce repeated textures, we only allow up to c shifted copies of an image to be used for texture transfer (typically $c = 2$).

Once the alignment step is over we have a set of ck shifted real images that we can now use for texture transfer. We model the problem as label assignment over a Markov Random field (MRF) and solve it using graph cuts. The set of labels at each pixel location consists of up to $ck + 1$ labels, corresponding to c shifted versions of each of the k real images, as well as a copy of the CG image, in case no matching texture was found. We look for the best label assignment to optimize an objective function $C(L)$ that consists of a data term C_d over all pixels p and an interaction term C_i over all pairs of pixels p, q. Specifically

$$C(L) = \sum_p C_d(p, L(p)) + \sum_{p,q} C_i(p, q, L(p), L(q)), \tag{3}$$

where the data penalty term $C_d(p, L(p))$ that measures distance between a 3×3 patch around pixel p in the CG image and a real image is given by

$$C_d(p, L(p)) = \alpha_1 D_c(p, L(p)) + \alpha_2 D_g(p, L(p)) + \alpha_3 D_l(p, L(p)). \tag{4}$$

The term $D_c(p, L(p))$ is the average distance in L*a*b* space between the 3×3 patch centered around pixel p in the CG image and the patch centered around pixel p in the image associated with label $L(p)$. Similarly, $D_g(p, L(p))$ is the average distance between the magnitudes of the gradients of the patches. Note that both of these terms are zero when the label $L(p)$ corresponds to the CG image.

The region label term $D_l(p, L(p))$ in (4) controls the error of transferring textures between different cosegmentation regions and also provides an error for choosing the original CG image as a source of texture:

$$D_l(p, L(p)) = \begin{cases} \gamma, & L(p) = \text{CGlabel}, \\ 0, & p \text{ and } L(p) \text{from same region}, \\ 1, & \text{otherwise}. \end{cases} \tag{5}$$

If the distance of all the real images is greater than $\alpha_3\gamma$, the graph cut algorithm will prefer keeping the CG texture.

The weights in (4) balance the contributions of the three terms and they are normalized, $\sum_i \alpha_i = 1$. The default values of 0.2, 0.7, and 0.1 give more weight to the gradient term than the other terms, but these weights can be adjusted by the user.

The interaction term $C_i(p, q, L(p), L(q))$ in (3) is zero if the labels $L(p)$ and $L(q)$ are the same, and a constant modulated by a blurred and inverted edge map of the CG image otherwise. Specifically

$$C_i(p, q, L(p), L(q)) = \begin{cases} 0, & L(p) = L(q), \\ \lambda M(p), & \text{otherwise}, \end{cases} \tag{6}$$

where $M(p)$ is zero near strong edges in the CG image and near 1 in smooth regions. The scalar λ affects the amount of texture switching that can occur. For low values of λ, the algorithm will prefer small patches of textures from many images and for high values of λ the algorithm will choose large blocks of texture from the same image. The typical range of this parameter is 0.1 to 0.4.

Once the graph cut has determined the label assignment per pixel (i.e., the image from which to transfer texture), we copy gradients from the selected image into the CG image and then reconstruct the image by solving Poisson's equation with Neumann boundary constraints. To minimize reconstruction errors near region boundaries, we use a mixed guidance field, selecting the gradient based on its norm [37].

An example of texture transfer from three real images to a CG image is shown in Fig. 6. The input CG image is shown in the upper left and two matches returned from our database are shown in the upper right; we allowed two shifted copies of each real image ($c = 2$). Gradients were copied from the real images into the CG image in regions specified by the graph cut. The resulting image, after color and tone adjustment, is shown in the lower left.

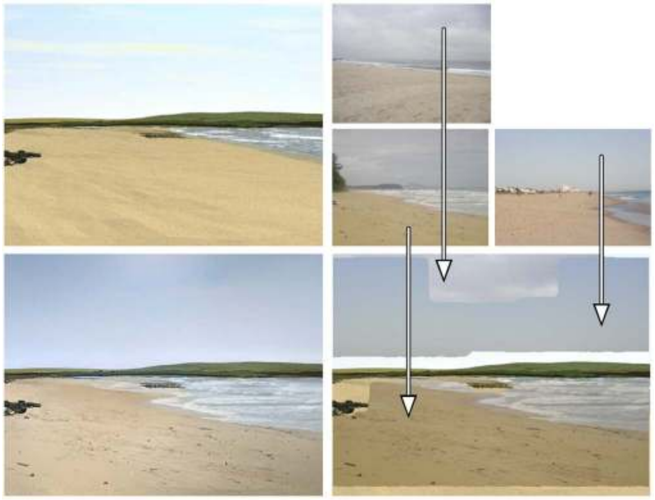


Fig. 6. CG image (upper left) with regions transferred from three reference photographs (upper right to lower right). The composite after Poisson blending and color and tone transfer (bottom left).

5 Results

Figs. 9 and 11 show results generated by our system for a number of different scenes. These examples include city scenes and various landscapes—e.g., forest, meadow, lake, and beach. To generate these results, we applied all three stages of our system, transferring texture first and then altering the color and tone of the input CG images to match the real images. We perform texture transfer first because gradient-domain manipulations can cause color shifts that deviate from natural colors. The color and tone transfer steps correct these unnatural colors by shifting them toward colors in the real images.

In our system, color and tone transfer operations are completely automatic. And for the examples shown in Fig. 11, texture transfer only requires the user to specify a few parameters. The most important parameters are γ and λ, described in Section 4.3. The parameter γ controls the preference of the algorithm between real and CG textures. Setting this parameter to a high value will cause the algorithm to recreate the CG image, as best it can, with only real patches. Setting it to a low value will cause the algorithm to only choose real patches if they are close to the CG patch, otherwise preserving the original patch. The λ parameter controls the size of the transferred texture patches. Because all transfer operations are reasonably fast, these parameters can be adjusted to synthesize different versions of an image.

As shown in Figs. 6 and 11, many natural scenes can be synthesized using an automatic approach. These scenes are primarily composed of stationary textures, and the global structure of the image is conveyed by the composition, boundaries, and scale of the textures within each image. For scenes with more structure, however, such as the examples in Figs. 7a and 7b, automatic texture transfer can fail. For these scenes, differences in perspective, scale, and scene geometry between the CG input and real matches can lead to objectionable artifacts in the final result. These problems are due to the fact that our MRF-based texture transfer makes local decisions, which are based solely on 2D image data. Therefore, properties of the image that are

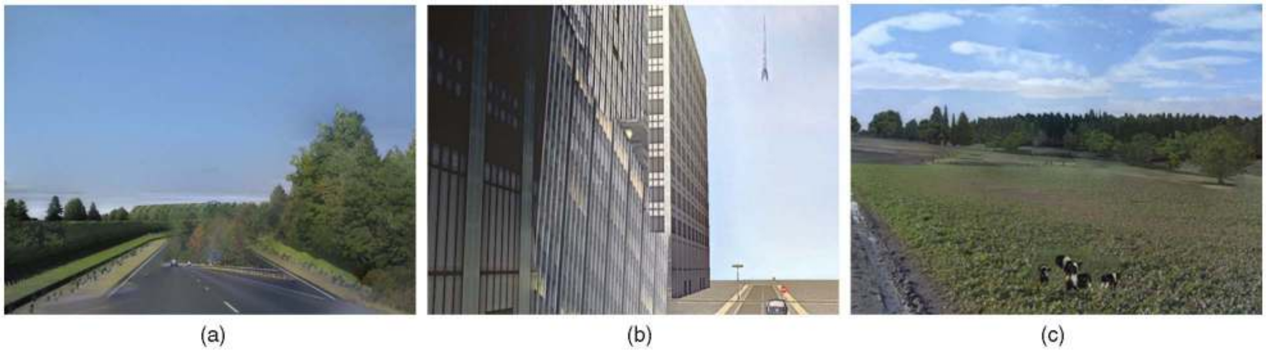


Fig. 7. Failure cases. (a) and (b) The texture transfer algorithm does not account for viewpoint or geometric differences between objects in the source images. As a result, images of structured scenes are difficult to blend without user interaction. (c) Differences in scale can also lead to unrealistic composites. The cows from the image in Fig. 4 appear to be the size of mice because the texture is at the wrong scale.

strongly dependent on the geometry of the original scene can be mishandled.

Fortunately, a few operations are sufficient to extend our method to a larger class of scenes, including those with significant structure. In particular, we can use the spidery mesh of Horry et al. [38] or simple planar homographies to correct major structural differences. Once warped, texture transfer proceeds as in Section 4.3.

In Fig. 9, the real images were warped to align geometry before texture transfer. In the top row, the road from a real image was warped using the spidery mesh tool to more closely match the road of the CG image. Trees, mountains, and the sky from other (unwarped) image matches were added to produce the final result. In the second row, textures of buildings were warped by specifying planes in both the real and CG images. The color and tone of the image was also modified to match a real image (not shown). In the bottom row, the user adjusted the blending mask within the red lines to preserve the dashed line in the center of the road.

In some cases, a user may not want a region of the image to be modified substantially. For example, if the user has spent hours editing a 3D model they probably want the model to appear exactly as they have designed it. In these cases, the user can specify an alpha mask and the system will operate outside the mask. In this scenario, the system provides an easy way to synthesize a realistic background for a 3D model. Two examples of 3D models with backgrounds synthesized by our system are shown in Fig. 8.

6 Evaluation

We conducted a user study to evaluate the effectiveness of our system. From a set of 10 example CG images, we generated 10 CG2Real results automatically, i.e., without manual adjustments. A third set of 10 images was selected from the real photographs used to enhance the corresponding CG2Real results.

Twenty subjects, between 20 and 30 years old, participated in the user study. They were paid $10 for their time. Each participant viewed a sequence of 10 images drawn from the set of 30. Each test sequence was selected randomly, with the constraint that the sequence contained at least three images from each category, and that multiple instances of images from the same example did not appear in the sequence. Participants were instructed to identify each image as "real" if they felt that the image was captured by a camera and "fake" if they felt it was generated by a computer program. They were also informed that their responses would be timed but that they should respond accurately rather than quickly.

Here, we report a number of findings. With unlimited viewing time, the subjects classified:

- 97 percent of the CG images as fake;
- 52 percent of the CG2Real images as fake;
- 17 percent of the real images as fake.

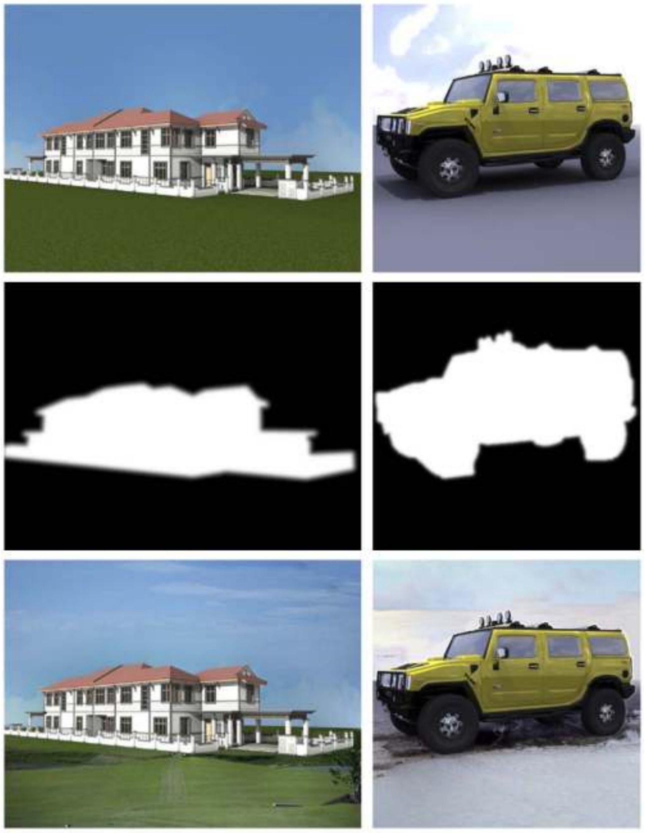


Fig. 8. Realistic backgrounds for CG models. In some cases, a user may not want regions of the image replaced. The user can specify an alpha mask (middle row) and the system will operate outside the mask, thus creating a realistic context for a rendered model (bottom).

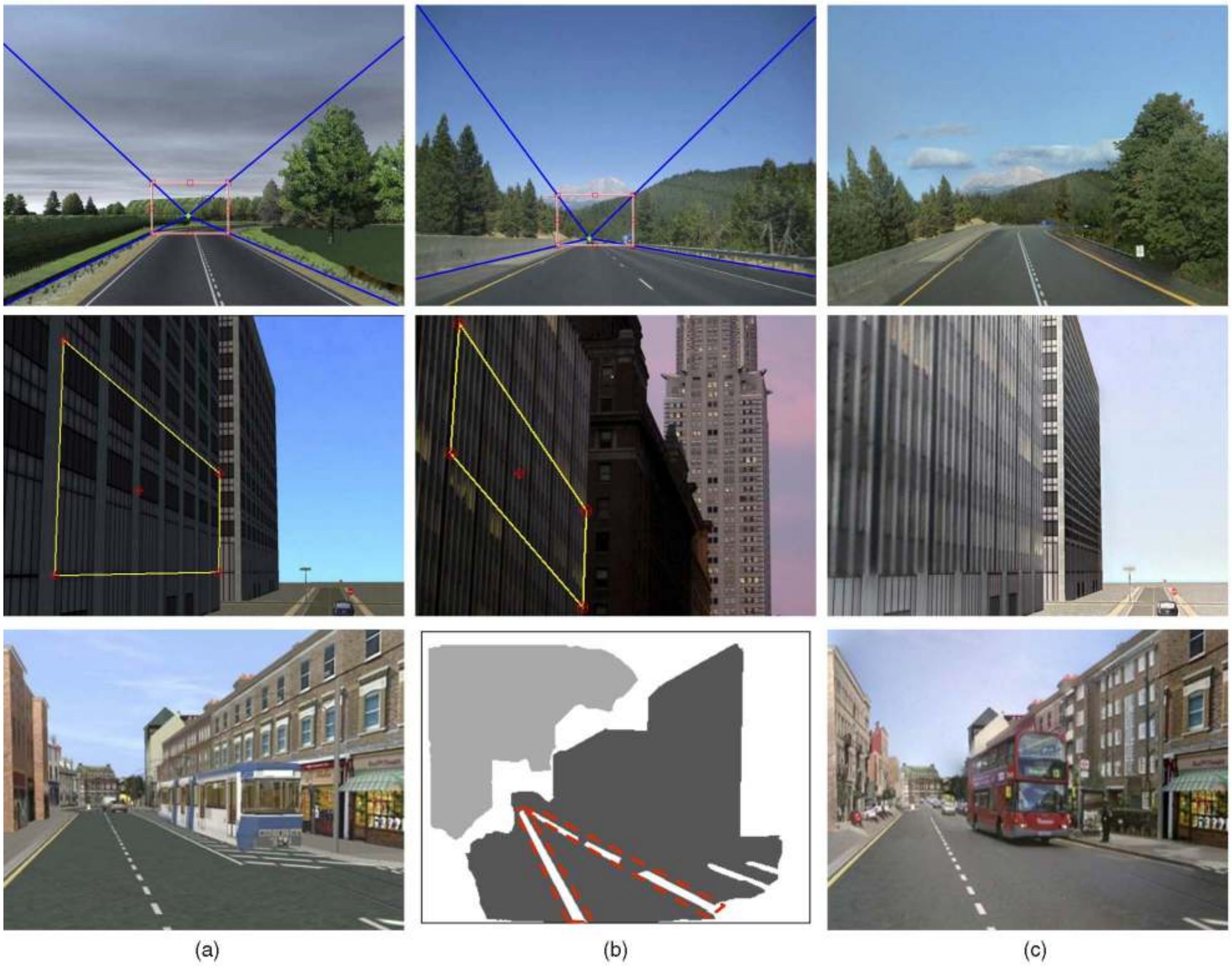


Fig. 9. CG2Real results on structured scenes. (a) Images of structured scenes are difficult to blend without user interaction. (b) The spidery-mesh tool (top) and rectangle tool (middle) were used to align geometry. In the last row of (b), a scribble tool was used to modify the mask produced by our texture transfer algorithm (the modifications are outlined). (c) Final results using edited textures.

As can be seen, users were able to identify 97 percent of the CG images as fake and and this number was reduced to 52 percent, indicating that some of the CG2Real images were considered real. With only 5 seconds of viewing, the subjects classified:

- 82 percent of the CG images as fake;
- 27 percent of the CG2Real images as fake;
- 12 percent of the real images as fake.

In this case, most of the CG images were still identified as fake, but more of the CG2Real images were considered real. These results suggest that our color, tone, and texture transfer give an initial impression of realism, but with unlimited viewing time, inconsistencies, such as scale and perspective differences, become apparent.

Fig. 10a shows the complete results. It describes the percentage of images marked as fake as a function of maximum response time.

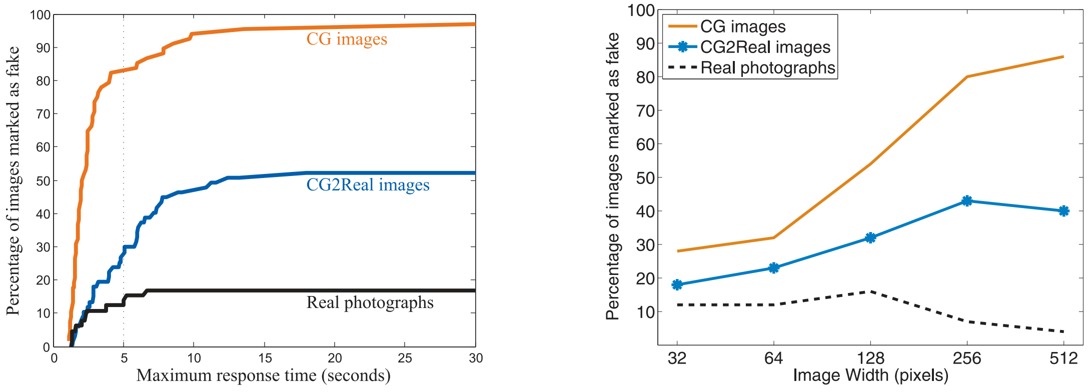
6.1 Realism versus Size

What cues are viewers using to distinguish CG from real? To begin to answer this question, we repeated the real versus fake discrimination task at various scales. By changing the size of the images, we change the amount of high-frequency information, and our goal was to quantify the effect of this information on the perception of realism.

We presented three sets of images (CG, CG2Real, and Real) and we fixed the presentation time at five seconds. We varied the image width in powers of two from 32 to 512 pixels and asked the viewer to identify the images as "real" or "fake" using the same definitions as the previous study.

We used the 30 images from the previous study (10 of each type) and collected 20 responses per image at five different sizes. Due to the size of this study ($30 \times 5 \times 20 = 3{,}000$ responses), we used Amazon's Mechanical Turk [39]. Mechanical Turk is an online marketplace for human intelligence tasks. Requestors can publish tasks and the rate they are willing to pay per task; workers can browse for tasks they are willing to perform. We divided the study into experiments by image size, collecting 20 responses for each image in each experiment. The average time per experiment was 12 minutes and the average number of contributing workers was 39.

The results are shown in Fig. 10b. At 32 pixels, most images were labeled as "real," though there was some ability to distinguish between the three types of images even at this scale. As the image size increased, the task became easier,



(a)

(b)

Fig. 10. (a) Percentage of images marked as fake as a function of maximum response time for real photos, CG images, and CG2Real images produced by our method. (b) Percentage of images marked as fake as a function of image width for real photos, CG images, and CG2Real images produced by our method.

though the ability to detect a CG image as fake increased more dramatically with size than the ability to detect a CG2Real image (based on the slope of the curves between 32 and 256 pixels). In addition, the viewer's confidence in real images increased after 128 pixels—they mislabeled fewer real images as fake. This study suggests that high frequencies contribute to the perception of realism in CG images, though they do not account for all of it.

In our system, we apply three types of transfer and the previous experiments have established that all three of these operations together improve realism for CG images. But how much realism is due to texture transfer and how much is due to color and tone transfer? To address this question, we ran another Mechanical Turk task showing images for five seconds and asking viewers to label the images as real or fake. For this experiment, we introduced CG2Real images without texture transfer (only color and tone), but kept the other parameters of the experiment the same. We found that 69 percent of color and tone images were classified as fake, which is between the results for CG and CG2Real (full pipeline) found in previous studies. This result suggests that color and tone transfer improve realism somewhat, but most of the gain shown in Figs. 10a and 10b comes from texture transfer.

6.2 Search Descriptors

As we mentioned in Section 3.1, we originally used the gist descriptor for image search but were not satisfied with the results. We found that the spatial pyramid descriptor consistently returned better matches, but we sought to quantify this observation. We used Mechanical Turk again for this task but with a different experimental setup.

We selected 50 CG images from our collection at random and performed queries on the same image database using two different descriptors: our spatial pyramid descriptor and gist. We hired 10 workers per image for a total of 1,000 tasks (50 inputs $\times$ 10 workers per input $\times$ 2 descriptors).

In a given task, we presented the user with a CG image and 20 image matches. Users were instructed to "select all images that were good matches to the query image." There were additional instructions to clarify that a good match is an image that depicts a similar scene. Users responded via checkboxes, and their responses were not timed.

Here, we consider a match "good" if three or more users (out of 10) considered it so. Across 50 images, at this 30 percent acceptance threshold, the spatial pyramid descriptor returned an average of 4.6 good matches per image, while gist returned 1.7 good matches per image.

7 Limitations

In general, our system is only as good as the database of photographs upon which it is built. We have found that a database of 4.5 million images yields around four good matching images for CG landscapes, but more images would be required to match more complex scenes. In future work, we hope to increase the database size to determine how the number of matching images and the match quality affect the final result. We also hope to expand the database to include a wider variety of photographs, including indoor images. But even with more images, texture transfer may fail to be realistic if there are geometric or perspective differences or if the scale is incorrect, Fig. 7c. Some of these examples can be improved with texture warping, Fig. 9, and it is possible that an appropriate warp could be determined automatically in some cases.

8 Conclusions

Our system takes a radically different approach to synthesizing photo-realistic CG images. Instead of adding geometric and texture details or using more sophisticated rendering and lighting models, we take an image-based approach that exploits the growing number of real images that are available online. We transfer realistic color, tone, and texture to the CG image and we show that these transfers improve the realism of CG images. To improve upon these results, future work will have to consider other operations, perhaps perspective and scale adjustments. While we have found that high frequencies in real-image texture contribute to perceived realism, what other cues are people using to distinguish CG from real? Are they high or low level? Future work on this problem could provide valuable insight into the mysteries of human vision.

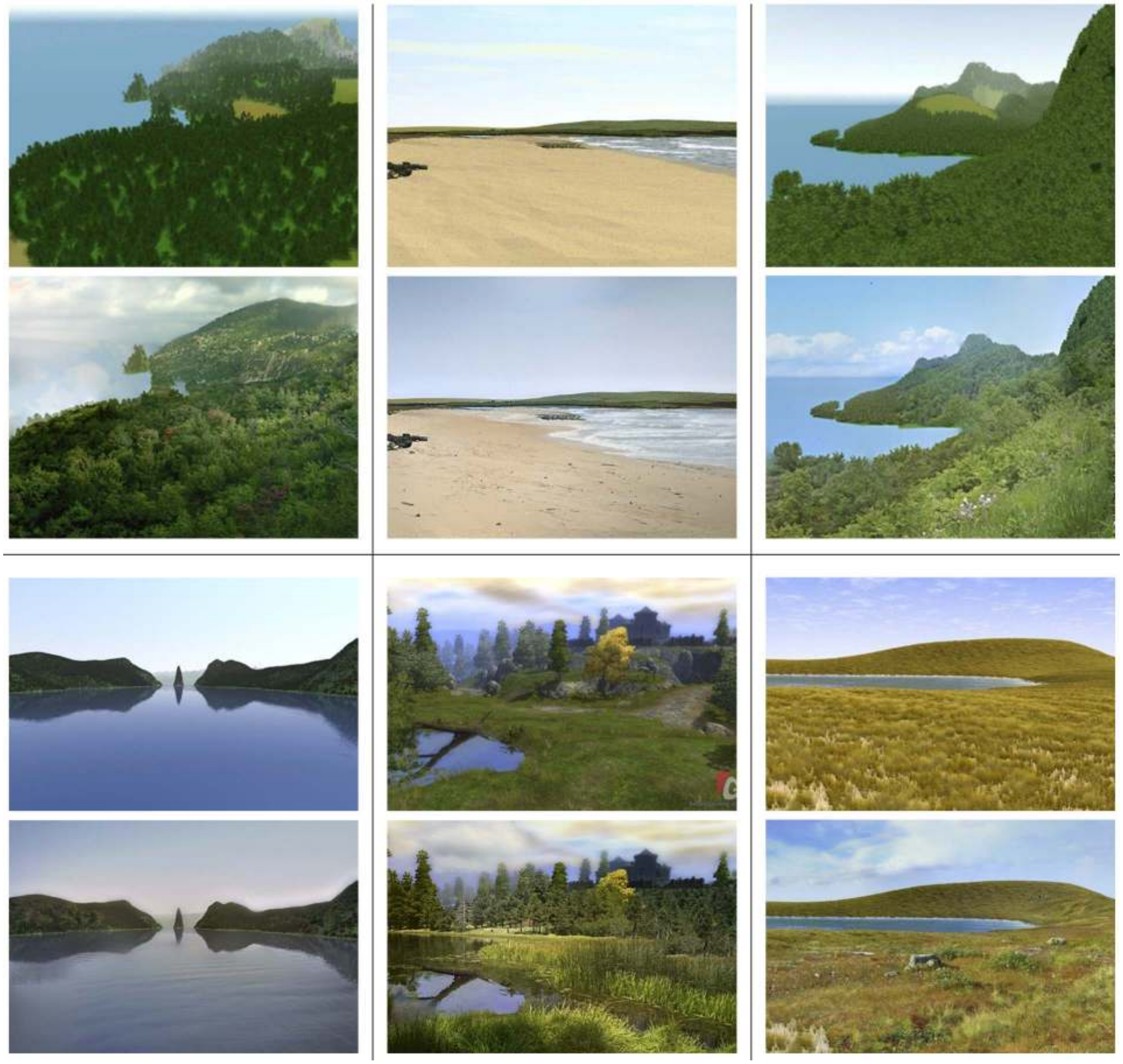


Fig. 11. Results of our CG2Real system. In each section, the CG input image appears above the output image. All of these results were synthesized without the use of scribbles or geometric adjustment.

Acknowledgments

This work was supported in part by the US National Science Foundation under Grants No. PHY-0835713 and 0739255, the John A. and Elizabeth S. Armstrong Fellowship at Harvard, and through generous support from Adobe Systems. The authors would like to thank David Salesin and Tom Malloy at Adobe for their continuing support. This work was completed while S. Avidan and W. Matusik were Senior Research Scientists at Adobe Systems.

References

[1] J. Hays and A.A. Efros, "Scene Completion Using Millions of Photographs," *ACM Trans. Graphics,* vol. 26, no. 3, pp. 1-7, Aug. 2007.

[2] D.J. Heeger and J.R. Bergen, "Pyramid-Based Texture Analysis/Synthesis," *Proc. ACM SIGGRAPH,* pp. 229-238, 1995.

[3] J.S. De Bonet, "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," *Proc. ACM SIGGRAPH,* pp. 361-368, 1997.

[4] A.A. Efros and T.K. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1033-1038, 1999.

[5] A.A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proc. ACM SIGGRAPH,* pp. 341-346, 2001.

[6] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 277-286, July 2003.

[7] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin, "Image Analogies" *Proc. ACM SIGGRAPH,* pp. 327-340, Aug. 2001.

[8] W.T. Freeman, T.R. Jones, and E.C. Pasztor, "Example-Based Super-Resolution," *IEEE Computer Graphics and Applications,* vol. 22, no. 2, pp. 56-65, Mar./Apr. 2002.

[9] R. Rosales, K. Achan, and B. Frey, "Unsupervised Image Translation," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 472-478, 2003.

[10] E. Reinhard, M. Ashikhmin, B. Gooch, and P.S. Shirley, "Color Transfer between Images," *IEEE Computer Graphics and Applications,* vol. 21, no. 5, pp. 34-41, Sept./Oct. 2001.

[11] F. Pitié, A.C. Kokaram, and R. Dahyot, "N-Dimensional Probability Density Function Transfer and Its Application to Colour Transfer," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1434-1439, 2005.

[12] S. Bae, S. Paris, and F. Durand, "Two-Scale Tone Management for Photographic Look," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 637-645, July 2006.

[13] C.-L. Wen, C.-H. Hsieh, B.-Y. Chen, and M. Ouhyoung, "Example-Based Multiple Local Color Transfer by Strokes," *Computer Graphics Forum,* vol. 27, no. 7, pp. 1765-1772, 2008.

[14] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng, "Intrinsic Colorization," *ACM Trans. Graphics,* vol. 27, no. 3, pp. 1-9, 2008.

[15] K. Dale, M.K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister, "Image Restoration Using Online Photo Collections," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 2217-2224, 2009.

[16] J.-F. Lalonde, D. Hoiem, A.A. Efros, C. Rother, J. Winn, and A. Criminisi, "Photo Clip Art," *ACM Trans. Graphics,* vol. 26, no. 3, pp. 1-10, Aug. 2007.

[17] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman, "Labelme: A Database and Web-Based Tool for Image Annotation," *Int'l J. Computer Vision,* vol. 77, nos. 1-3, pp. 157-173, May 2008.

[18] M. Johnson, G.J. Brostow, J. Shotton, O. Arandjelovic, V. Kwatra, and R. Cipolla, "Semantic Photo Synthesis," *Computer Graphics Forum,* vol. 25, no. 3, pp. 407-414, Sept. 2006.

[19] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, "Sketch2Photo: Internet Image Montage," *ACM Trans. Graphics,* vol. 28, no. 5, pp. 1-10, 2009.

[20] S. Lyu and H. Farid, "How Realistic is Photorealistic?," *IEEE Trans. Signal Processing,* vol. 53, no. 2, pp. 845-850, Feb. 2005.

[21] J.-F. Lalonde and A.A. Efros, "Using Color Compatibility for Assessing Image Realism," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1-8, 2007.

[22] Y. Weiss and W.T. Freeman, "What Makes a Good Model of Natural Images?" *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1-8, 2007.

[23] S. Roth and M.J. Black, "Fields of Experts: A Framework for Learning Image Priors," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 860-867, 2005.

[24] B.C. Russell, A.A. Efros, J. Sivic, W.T. Freeman, and A. Zisserman, "Segmenting Scenes by Matching Image Composites," *Proc. Advances in Neural Information Processing Systems,* pp. 1580-1588, 2009.

[25] A. Oliva and A. Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope," *Int'l J. Computer Vision,* vol. 42, no. 3, pp. 145-175, 2001.

[26] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study," *Int'l J. Computer Vision,* vol. 73, no. 2, pp. 213-238, 2007.

[27] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[28] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 2169-2178, 2006.

[29] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 1458-1465, 2005.

[30] Y.-W. Tai, J. Jia, and C.-K. Tang, "Local Color Transfer via Probabilistic Segmentation by Expectation-Maximization," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 747-754, 2006.

[31] C. Rother, T. Minka, A. Blake, and V. Kolmogorov, "Cosegmentation of Image Pairs by Histogram Matching-Incorporating a Global Constraint into MRFs," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 993-1000, 2006.

[32] K. Fukunaga and L.D. Hostetler, "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition," *IEEE Trans. Information Theory,* vol. 21, no. 1, pp. 32-40, Jan. 1975.

[33] S. Paris and F. Durand, "A Topological Approach to Hierarchical Segmentation Using Mean Shift," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1-8, 2007.

[34] J. Kopf, M.F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint Bilateral Upsampling," *Proc. ACM SIGGRAPH,* 2007.

[35] E.H. Adelson, E.P. Simoncelli, and R. Hingorani, "Orthogonal Pyramid Transforms for Image Coding," *Proc. SPIE Visual Comm. and Image Processing II,* vol. 845, pp. 50-58, 1987.

[36] Y. Li, L. Sharan, and E.H. Adelson, "Compressing and Companding High Dynamic Range Images with Subband Architectures," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 836-844, 2005.

[37] P. Pérez, M. Gangnet, and A. Blake, "Poisson Image Editing," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 313-318, July 2003.

[38] Y. Horry, K.-I. Anjyo, and K. Arai, "Tour into the Picture: Using a Spidery Mesh Interface to Make Animation from a Single Image," *Proc. ACM SIGGRAPH,* pp. 225-232, 1997.

[39] Amazon.com, "Amazon Mechanical Turk," http://www.mturk.com, 2009.

Micah K. Johnson received the undergraduate degrees in mathematics and music theory from the University of New Hampshire, the AM degree in electro-acoustic music from Dartmouth College, and the PhD degree in computer science from Dartmouth College. He is a postdoctoral fellow at MIT. His PhD work focused on image forensics, i.e., detecting tampered images, but he has since worked on a variety of techniques for improving realism in computer graphics. He is a member of the IEEE.

Kevin Dale received the BS degree in computer science from the University of North Carolina at Chapel Hill in 2004 and the MCS degree from the University of Virginia in 2006. He is currently, a PhD student at Harvard University. His research interests are in computational photography and graphics hardware. He is a student member of the IEEE.

Shai Avidan received the PhD degree from the School of Computer Science at the Hebrew University, Jerusalem, Israel, in 1999. He is a professor at Tel-Aviv University, Israel. He was a postdoctoral researcher at Microsoft Research, a project leader at MobilEye, a research scientist at Mitsubishi Electric Research Labs (MERL), and a senior research scientist at Adobe Systems. He has published extensively in the fields of object tracking in video sequences and 3D object modeling from images. Recently, he has been working on the Internet vision applications such as privacy preserving image analysis, distributed algorithms for image analysis, and media retargeting, the problem of properly fitting images and video to displays of various sizes. He is a member of the IEEE.

Hanspeter Pfister received the PhD degree in computer science from the State University of New York at Stony Brook. He is the Gordon McKay professor of the practice in Harvard University's School of Engineering and Applied Sciences. His research interests are at the intersection of visualization, computer graphics, and computer vision. He is a senior member of the IEEE and the IEEE Computer Society and member of the ACM, the ACM Siggraph, and the Eurographics.

William T. Freeman received the PhD degree from the Massachusetts Institute of Technology in 1992. He is a professor of electrical engineering and computer science at the Massachusetts Institute of Technology (MIT), working in the Computer Science and Artificial Intelligence Laboratory (CSAIL). He has been on the faculty at MIT since 2001. From 1992 to 2001, he worked at Mitsubishi Electric Research Labs (MERL), Cambridge, Massachusetts. Prior to that, he worked at Polaroid Corporation, and in 1987-1988, was a foreign expert at the Taiyuan University of Technology, China. His research interests include machine learning applied to problems in computer vision and computational photography. He is a fellow of the IEEE.

Wojciech Matusik received the BS degree in electrical engineering and computer science from the University of California at Berkeley in 1997, the MS degree in electrical engineering and computer science from MIT in 2001, and the PhD degree in 2003. He is a senior research scientist at Disney Research Zurich. Before coming to Disney, he worked at Mitsubishi Electric Research Laboratories and at Adobe Systems. His primary research is in the area of computer graphics with broad applications in other disciplines such as digital communications, materials science, and biomechanics.