Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2009-034 July 15, 2009

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

Micah K. Johnson, Kevin Dale, Shai Avidan, Hanspeter Pfister, William T. Freeman, and Wojciech Matusik

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

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

Micah K. Johnson[1,2] Kevin Dale[3] Shai Avidan[2] Hanspeter Pfister[3] William T. Freeman[1] Wojciech Matusik[2]

[1]MIT {kimo, billf}@mit.edu

[2]Adobe Systems, Inc. {avidan, wmatusik}@adobe.com

[3]Harvard University {kdale,pfister}@seas.harvard.edu

Figure 1: *Given an input CG image (left), our system finds the most similar photographs to the input image (not shown). Next, it 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).*

Abstract

Computer Graphics (CG) has achieved a high level of realism, producing strikingly vivid images. This realism, however, comes at the cost of long and often expensive manual modeling, and most often humans can still distinguish between CG images and real images. We present a novel method to make CG images look more realistic that is simple and accessible to novice users. Our system 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 novel 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 improved CG images appear more realistic than the originals.

1 Introduction

The field of image synthesis has matured to the point where photorealistic Computer Graphics (CG) images can be produced with commercially available software packages (e.g., Renderman and POVRay). However, reproducing the details and quality of a natural image often requires a considerable investment of time by a highly skilled artist. Even with large budgets and many man-hours of work, it is still often surprisingly easy to distinguish CG images from photographs.

CG images differ from real photographs in three major ways. First, the color distribution of CG images is often overly saturated and exaggerated. Second, multi-scale image statistics, such as the histogram of filter outputs at different scales, rarely match the statistics of natural images. Finally, CG images often lack details (i.e., high frequencies, texture, and noise) that make them look too pristine.

Recently, the proliferation of images available online through photo-sharing sites such as Flickr has allowed researchers to collect large databases of natural images and to develop data-driven methods for improving photographs. This work leverages a large collection of images to improve the realism of computer generated images in a data-driven manner with minimal user input.

CG2Real takes a CG image to be improved, retrieves and aligns a small number of similar natural images from a database, and transfers the color, tone, and texture from the natural images to the CG image. A key ingredient in the system is a novel mean-shift cosegmentation algorithm that matches regions in the CG image with regions in the real images. If the structure of a real image does not fit that of the CG image (e.g., because of differences in perspective), we provide a user interface to correct basic perspective differences interactively. After cosegmentation, we use local style transfer between image regions, which greatly improves the quality of these transfers compared to global transfers based on histogram matching. The user has full control over which regions and which styles are being transferred. 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 a minute for 600×400 pixel image.

The primary contribution of this paper is a novel data-driven approach for improving the look of CG images using real photographs. Within this system, several novel 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 improved CG images appear more realistic than the originals.

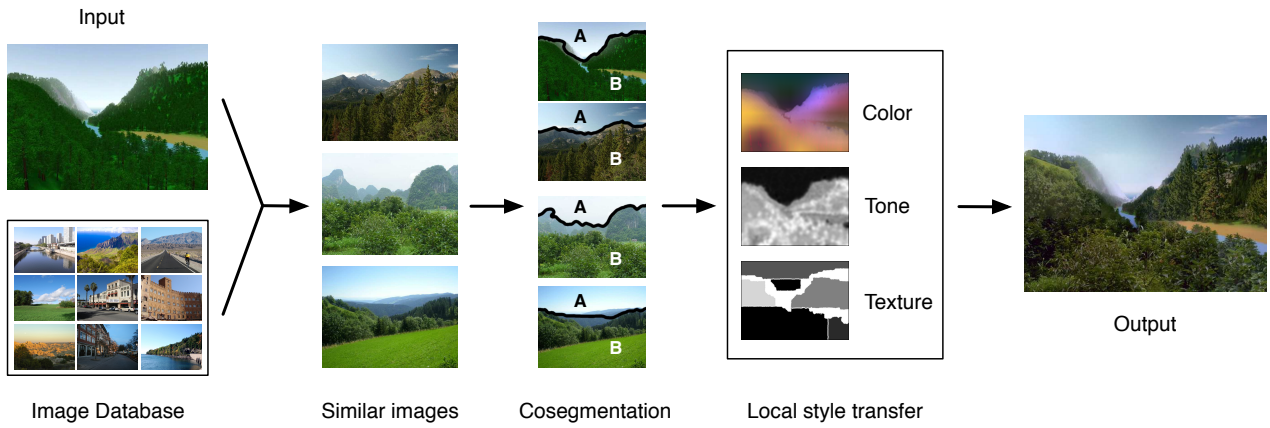

Figure 2: *An overview of our system. We start by querying a large collection of photographs to retrieve the most similar images. The user selects the k closest matches and the images, both real and CG, are cosegmented to identify similar regions. Finally, the real images are used by the local style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

2 Previous Work

Adding realistic texture to an image is an effective tool to improve the photo-realistic look and feel of CG images. In their seminal work, Heeger and Bergen [1995] proposed a novel texture synthesis approach. 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 [1997] to use joint multi-scale statistics. Alternatively, one can take an exemplar based approach to texture synthesis. This idea was first illustrated in the work of Efros and Leung [1999] and was later extended to work on patches instead of pixels [Efros and Freeman 2001; Kwatra et al. 2003]. The image analogies framework [Hertzmann et al. 2001] extends non-parametric texture synthesis by learning a mapping between a given exemplar pair of images and applying the mapping to novel images. Freeman et al. [2002] proposed a learning-based approach to solve a range of low-level image processing problems (e.g., image super-resolution) 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. [2003] later 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) in order to learn a mapping, one can take a global approach that attempts to transfer color or style between images. Reinhard et al. [2001] modified the color distribution of an image to give it the look and feel of another image. They showed results on both photographic and synthetic images. Alternatively, Pitié et al. [2005] consider this problem as estimating a continuous N-dimensional transfer function between two probability distribution functions and present an iterative non-linear algorithm. Bae et al. [2006] take yet another approach, using a two-scale nonlinear decomposition of an image to transfer style between images. In their approach, histograms of each layer are modified independently and then recombined to obtain the final output image. Finally, Wen et al. [2008] provide a stroke-based interface for performing local color transfer between images. In this system, the user provides a target image and input in the form of stroke pairs.

We build on and extend this line of work with several important distinctions. First, the work discussed so far does not consider the question of how the model images are chosen, and instead it is assumed that the user provides them. However, we believe a system capable of handling a variety of types of input images should be able to obtain model images with a minimum of user assistance. Given a large collection of photographs we assume that we can find images with similar global structure (e.g., trees next to mountains below a blue sky) and transfer their look and feel to the CG image. Moreover, 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. In one instance, Hays and Efros. [2007] use a large collection of images to complete missing information in a target image. The system works by retrieving a number of images that are similar to the query image and using their data 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. [2008] perform example-based image colorization using images from the web that is robust to illumination differences. However their method involves image registration between search results and input and requires exact scene matches. Our approach instead uses a visual search based on image data, and our transfers only assume similar content between CG input and real search results. Finally, Sivic et al. [2008] show a novel use of large image collections by retrieving and stitching together images that match a transformed version of the query real image. While not related to image editing, this work provides a unique image-browsing experience.

In work based on annotated image datasets, Lalonde and Efros [2007] use image regions drawn from the LabelMe database [Russell et al. 2008] to populate the query image with new objects. Johnson et al. [2006] allow the user to create novel composite images by typing in a few nouns at different image locations. Here the user input is used to retrieve and composite relevant parts of images from a large annotated image database. In both cases, the system relies on image annotations to identify image regions and region correspondences. In contrast, our approach uses an automatic cosegmentation algorithm for identifying local regions and inter-region correspondences.

Researchers have also studied the characteristics of natural versus synthetic images. For example, in digital forensics, Lyu and

Farid [2005] examine high order image statistics to distinguish between synthetic and natural images. Lalonde and Efros [2007] 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 [Weiss and Freeman 2007; Roth and Black 2005]. 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

Figure 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. Since high-quality CG images often have realistic textures, we typically transfer only color and tone for these inputs.

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 1024 pixels, was downsampled to approximately 75% its original size and stored in PNG format (24-bit color) to minimize the impact of JPEG compression artifacts on our algorithms.

3.2 Visual Search

The goal of the visual search is to retrieve semantically similar images for a CG input. For example, for a CG image depicting a park with a tree line on the horizon, the results of the query should depict similar scenes at approximately the same scale, with similar lighting, viewpoint, and spatial layout of objects within the image (e.g., a park with trees and a skyline). Using a very large database of real photographs greatly enhances the chances of finding good matches. Searching a large database, however, requires an efficient, yet descriptive, image representation.

The gist scene descriptor [Oliva and Torralba 2001] is one choice of representation that has been used successfully for image matching tasks [Hays and Efros 2007]. 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 out-performed by SIFT-based descriptors for texture classification [Zhang et al. 2007], justifying our decision to use a more detailed image representation.

Our representation is based on visual words, or quantized SIFT features [Lowe 1999], and the spatial pyramid matching scheme of Lazebnik et al. [2006]. This approach has been shown to perform well for semantic scene classification. Although our transfer operations are local, the system benefits from global structural alignment between the CG input and real matches, justifying a descriptor with significant spatial resolution. Additionally, since textures in the original CG image often only faintly resemble the real-world appearance of objects they represent, we use smaller visual word vocabularies than is typical to more 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 1×1, 2×2, 4×4, and 8×8, for the 50-word vocabulary, for a final pyramid descriptor with 4346 elements. This representation has some redundancy, since a visual word will occur multiple time across pyramid levels. The weighting scheme specified by the pyramid match kernel [Grauman and Darrell 2005] accounts for this; it also effectively provides term-frequency (tf) weighting. We also apply inverse document frequency (idf) weighting to the pyramid descriptor.

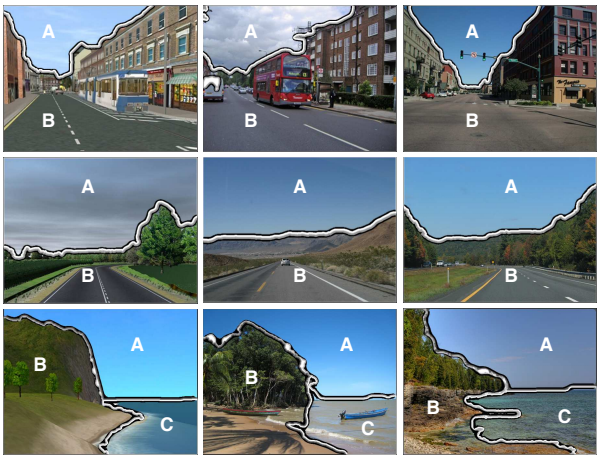

Figure 3: *Results from our cosegmentation algorithm. In each row, the CG image is shown on the left and two real image matches, on the right. Note that in all cases, segment correspondences are correct, and the images are not over-segmented.*

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.

3.3 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. [2006], 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 [Rother et al. 2006] 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 the cosegmentation process and matching regions are automatically produced as a byproduct.

CG input | Color model | Global histogram matching | Local color transfer

Figure 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, and our local color transfer algorithm. In the top example, the global result has a bluish color cast. In the bottom exmaple, the global result swaps the colors of the building and the sky. Local color transfer yields better results in both examples.*

The cosegmentation approach of Rother et al. [2006] 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 Rother et al., we take a simpler approach. Building upon the mean-shift framework [Fukunaga and Hostetler 1975], we define a new feature vector with color, spatial, and image-index terms. We can compute reasonable cosegmentations in seconds using a standard mean-shift implementation.

Our feature vector at every pixel p 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 j^{th} 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 and do not introduce an artificial distance along this dimension. In addition, the components of the feature vector are weighted by three weights to balance the color, spatial, and index components. 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 [Paris and Durand 2007]. 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 [Kopf et al. 2007].

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 non-sky. 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 over-segmented.

4 Local Style Transfer Operators

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

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 multi-dimensional 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 sub-sampled 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. [2005] 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

CG input | Tone model | Local color and tone transfer | Close-up

Figure 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.*

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.

4.1 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. [2006] 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 4-level pyramid based on a quadrature mirror filter [Adelson et al. 1987] 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 1-D 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:

$$s_i'(p) = g_i(p)s_i(p), \tag{1}$$

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. [2005] 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 Figure 5. As can be seen, the look and feel 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 photorealistic.

4.2 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. Simply working on a region-by-region basis, as was done for color and tone transfer did not work well. This is 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 overcome these challenges, we take a more global approach. First, we generate multiple, shifted copies of each real image and align them with the CG image based on strong-edge maps. Then we transfer the texture using graph-cut. The result is a coherent texture transfer that respects strong scene structure.

We perform the 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:

$$C(L) = \sum_p C_d(p,L(p)) + \sum_{p,q} C_i(p,q,L(p),L(q)) \tag{2}$$

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:

$$C_d(p,L(p)) = \alpha_1 D_c(p,L(p)) + \alpha_2 D_g(p,L(p)) + D_l(p,L(p)) \ . \tag{3}$$

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

In Eqn. 3, $D_c(p,L(p))$ 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 real image $L(p)$. Likewise, the term $D_g(p,L(p))$ is the average distance between the magnitudes of the gradients of the patches, and the term $D_l(p,L(p))$ is the region label term that is set to 0 if the CG patch and real patch are in the same segmented region, a constant (in our case we fix it to be 0.3) when the patches are from different regions, and a value in between for patches that span boundaries. The value of $C_d(p,L(p))$ is set to a constant when $L(p)$ corresponds to the CG label. This way, if the distance of all real images from the CG image is above that constant, the graph-cut algorithm will prefer keeping the CG texture.

The interaction term $C_i(p,q,L(p),L(q))$ is zero if the labels $L(p),L(q)$ are the same, and a constant (modulated by its distance from a strong edge in the CG image) otherwise. Specifically:

$$C_i(p,q,L(p),L(q)) = \begin{cases} 0 & L(p) = L(q) \\ M(p) & \text{otherwise} \end{cases} \quad (4)$$

where $M(p)$ is a distance transform mask computed around the largest gradients in the CG image (e.g., 10%). This helps preserve the large structures in the CG image.

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 [Pérez et al. 2003].

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.

5 User Interaction

In our system, color and tone transfer operations are completely automatic. And for the examples shown in Fig. 1 and Fig. 9, texture transfer only requires the user to specify a few parameters. The first parameter sets the percentage of large gradients to preserve in the CG image, typically 5–10%. We found that without this parameter, the texture transfer algorithm would often synthesize images that, while realistic, did not sufficiently resemble the CG image. The second parameter is the assigned cost of choosing a CG patch. It adjusts 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, while setting it to a low value will cause the algorithm to only choose real patches that are very close to the CG patch. Typical values of this parameter range from 0.2 to 0.5 since the data term C_d in Eqn. 3 often lies in the range 0 to 1. Because all transfer operations are reasonably fast, these parameters can be adjusted to synthesize different versions of an image.

As shown in Figs. 1, 6 and 9, 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 those in Fig. 7, 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 global properties of the image, such as symmetry, and properties that are 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. These operations allow the user to specify locally the importance of different regions of the CG and real images, as well as correct for differences in perspective and planar geometry.

Scribble interface. Here the user can indicate with a scribble regions from real image matches that should not be chosen during texture transfer. This is useful when, for example, only part of the real image is a good structural match. Likewise, the user can also scribble on the CG image to specify regions that should be protected. This allows the user to ensure that important, novel structure in the CG image not well-represented in the real image matches is maintained.

These scribbles are easily incorporated into texture transfer. For a scribble marking out a region of a real image, we simply set $C_d(p,L(p))$ to infinity for all pixels p under the scribble when $L(p)$ corresponds to the marked image. Similarly, for scribbles protecting regions of the CG image, we set $C_d(p,L(q))$ to zero for pixels q beneath the scribble when label $L(q)$ corresponds to the CG image.

Geometry interface. Here we focus on tools for adjusting the perspective, and to a lesser degree, the scale and orientation, of approximately planar regions. In the simplest case, the user specifies the vertices of corresponding rectangles in the CG and real images. From these rectangles, we compute the homography that maps the real image to the CG image and then warp accordingly. Once warped, texture transfer proceeds as in Sec. 4.2.

For corridor scenes, or those that can be approximated as such, we use the spidery mesh of Horry et al. [1997] to specify coarse scene geometry. The spidery mesh consists of a background rectangle and vanishing point, both of which are interactively positioned by the user. Together they determine five 3D rectangles that capture the coarse geometry of the scene. Once the user specifies geometry for CG and real images, we compute a homography for each rectangle in a real image that maps it to the corresponding rectangle in the CG image and warp to align real to CG. As above, the warped real image is then used for texture transfer.

6 Results

Figures 9 and 10 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 (the reviewers are

Figure 7: *Failure cases. The texture transfer algorithm does not account for geometric differences between objects in the source images. As a result, images of structured scenes are difficult to blend without user interaction. See Fig. 10 for examples of tools for improving these images.*

encouraged to look at the supplementary material for additional examples). To generate these results, we applied all three stages of our system, transferring color, tone, and texture to the input CG images. Each of these stages modified a different aspect of the CG image and increased its photorealism. We have observed that when the input images have low complexity then the region transfer greatly improves the realism of the image. However, when the input images are already complex, the color and tone adjustment steps are sufficient to improve realism.

The results in Fig. 9 were obtained using the automatic approach, where we selected the k best real images (typically two or three), from which our system synthesized the result. With a few adjustments to the parameters to balance real vs. CG textures, we were able to create reasonable results.

In Fig. 10, the challenging examples from Fig. 7 were edited via the geometry and scribble interfaces to produce more compelling results. 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 third row, buildings were warped using the spidery mesh tool to match the buildings in the CG image. However even after warping, there was an inconsistency that the graph-cut could not solve—there were multiple sets of dashed lines on the street. This error was easily fixed using the scribble interface to block the lines from the real images and allow those from the CG image to propagate to the final result.

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. 11.

7 Evaluation

We conducted a user study in order to quantitatively evaluate the effectiveness of our system. For a set of 10 example CG images, we generated 10 CG2Real results. Note that all results used in the study were generated without the use of scribbles or geometric adjustment. We did, however, adjust the two parameters discussed in Sec. 5 to create the most compelling result for each input. A third set of 10 images was selected from the real photographs used to enhance the corresponding CG2Real results. Each of 20 participants viewed a sequence of 10 images drawn from this set, with a total 30 images from 3 categories. Each test sequence was selected randomly, with the constraint that the sequence contain at least 3 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:

- 17% of the subjects classified real images as fake;
- 52% of the subjects classified CG2Real images as fake; and
- 97% of the subjects classified CG images as fake.

As can be seen, our system improved the "realism" of CG images by 45%. After 5 seconds of viewing:

- 12% of the subjects classified real images as fake;
- 27% of the subjects classified CG2Real images as fake; and
- 82% of the subjects classified CG images as fake.

In this case we have improved the "realism" of CG images by 55%. Figure 8 shows the complete results. It describes the percentage of images marked as fake as a function of maximum response time.

What cues are viewers using to distinguish CG from real? To begin to answer this question, we repeated the real vs. 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 as the previous study (10 of each type) and collected 20 judgements per image at 5 different sizes. Due to the size of this study ($30 \times 5 \times 20 = 3000$ responses), we used Amazon's Mechanical Turk [Amazon.com 2009]. 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 twelve minutes and the average number of contributing workers was thirty-nine.

The results are shown in Fig. **??**. At 32 pixels, most images were labelled 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, 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 mislabelled 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% 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. 8 and **??** comes from texture transfer.

7.1 Search descriptors

As we mentioned in section 3.2, we originally used the gist descriptor for image search but were not satisfied with the results. Our experience has been that the spatial pyramid descriptor consistently returns 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 at random from a working set of 120 CG images obtained via Google Images and performed queries on two separate databases: one using the spatial pyramid descriptor and one using gist. We hired 10 workers per image for a total of 1000 tasks (50 inputs $\times$ 10 workers per input $\times$ 2 match sets).

In a given task, we presented the user with a CG image and twenty 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 responsed via checkboxes, and their responses were not timed.

Here we consider a match "good" if 3 or more users (out of 10) considered it so. Across 50 images, at this 30% 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.

Note that the CG images were randomly selected; there were several that could not be expected to have similar matching images. In general, our approach is only as good as the database of photographs upon which it is built, and moreover, can only identify appropriate matches for input CG images of sufficient quality. However, the user can manually add images that resemble the CG scene or that they would like to use for transfers.

8 Conclusions

Our system takes a radically different approach to synthesizing photorealistic 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 by 45%. To improve upon these results, future work will have to consider other operations, perhaps perspective and scale adjustments. But even the best CG often lacks realism, so what cues are people using to distinguish CG from real? Are they high or low-level? This is a human vision question with implications beyond computer graphics and future work on this problem could provide valuable insight.

Our system is easy to use, offers some artistic control, and can be integrated into the visual content creation tool chain. It will benefit novice users, who do not have the time, skill, or access to high end modeling tools. At the same time it can serve professional artists that would like to add a photorealistic touch to their CG images.

Figure 11: *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 and the system will operate outside the mask, thus creating a realistic context for a rendered model.*

References

ADELSON, E. H., SIMONCELLI, E. P., AND HINGORANI, R. 1987. Orthogonal pyramid transforms for image coding. In *Proc SPIE Visual Communications and Image Processing II*, vol. 845, 50–58.

AMAZON.COM, 2009. Amazon mechanical turk. `http://www.mturk.com`.

BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. *ACM Transactions on Graphics 25*, 3 (July), 637–645.

DE BONET, J. S. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 361–368.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 341–346.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, 1033–1038.

FREEMAN, W. T., JONES, T. R., AND PASZTOR, E. C. 2002. Example-based super-resolution. *IEEE Computer Graphics & Applications 22*, 2 (Mar./Apr.), 56–65.

FUKUNAGA, K., AND HOSTETLER, L. D. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. on Info. Theory 21*, 1, 32–40.

GRAUMAN, K., AND DARRELL, T. 2005. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*.

HAYS, J., AND EFROS, A. A. 2007. Scene completion using millions of photographs. *ACM Transactions on Graphics 26*, 3 (July), 4:1–4:7.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 229–238.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 327–340.

HORRY, Y., ANJYO, K.-I., AND ARAI, K. 1997. Tour into the picture: using a spidery mesh interface to make animation from a single image.

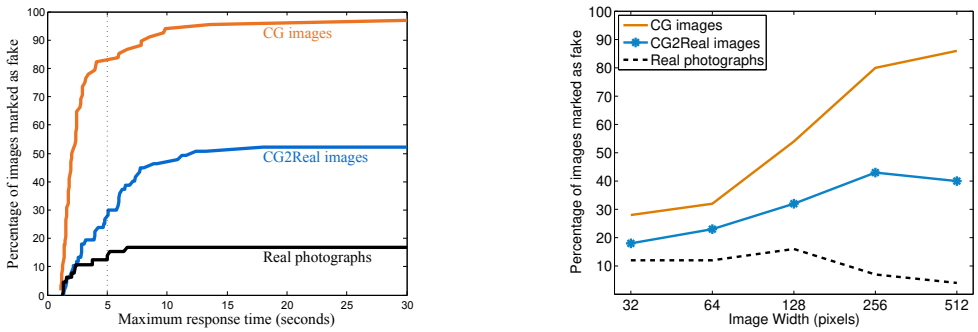

Figure 8: *(a) Percentage of images marked as fake as a function of maximum response time for real photos, CG images, and CG2Real images produced by our method. (b) Percentage of images marked as fake as a function of image width for real photos, CG images, and CG2Real images produced by our method.*

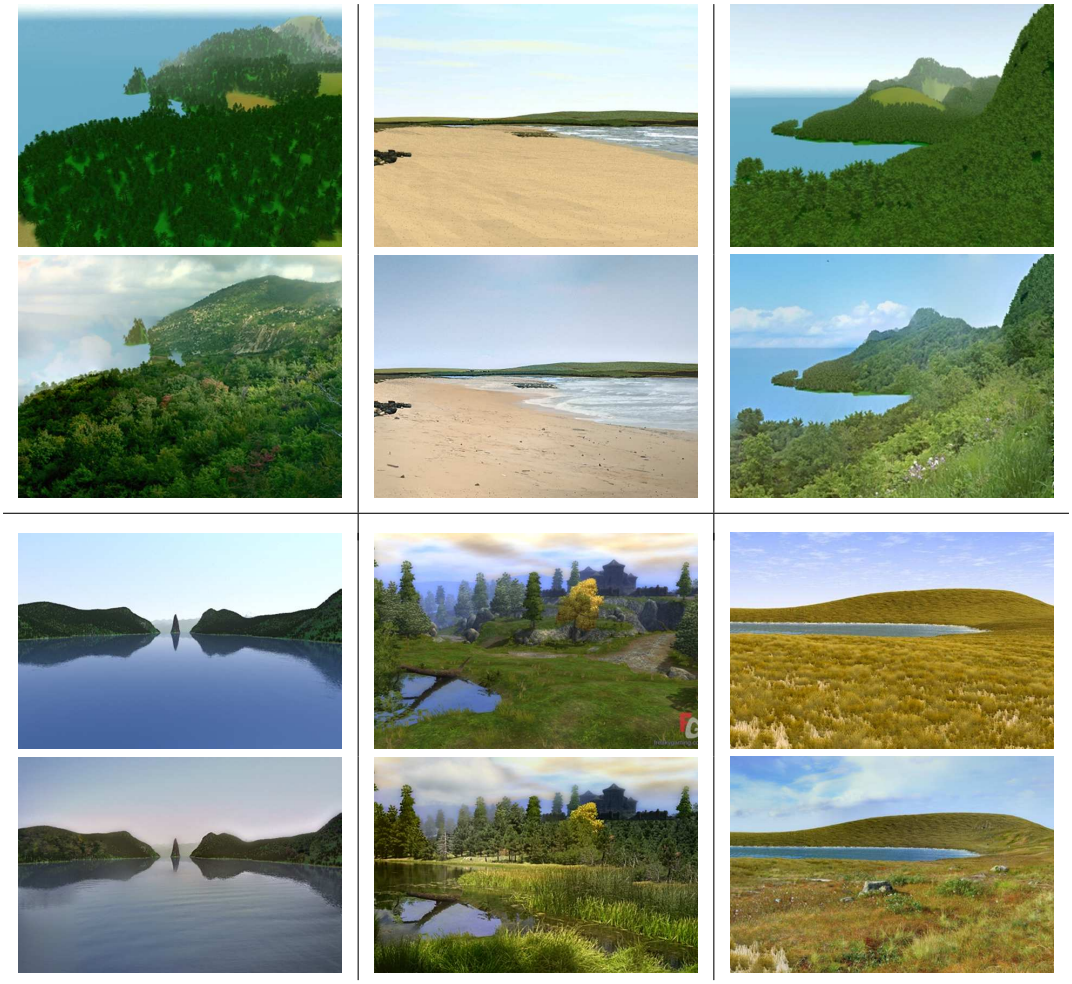

Figure 9: *Results of our CG2Real system. In each section, the CG input image appears above the output image. All of these results were synthesized without the use of scribbles or geometric adjustment.*

In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 225–232.

JOHNSON, M., BROSTOW, G. J., SHOTTON, J., ARANDJELOVIC, O., KWATRA, V., AND CIPOLLA, R. 2006. Semantic photo synthesis. *Computer Graphics Forum 25*, 3 (Sept.), 407–414.

KOPF, J., COHEN, M. F., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH) 26*, 3.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3 (July), 277–286.

LALONDE, J.-F., AND EFROS, A. A. 2007. Using color compatibility for assessing image realism. In *Proc. IEEE Int. Conf. Computer Vision*.

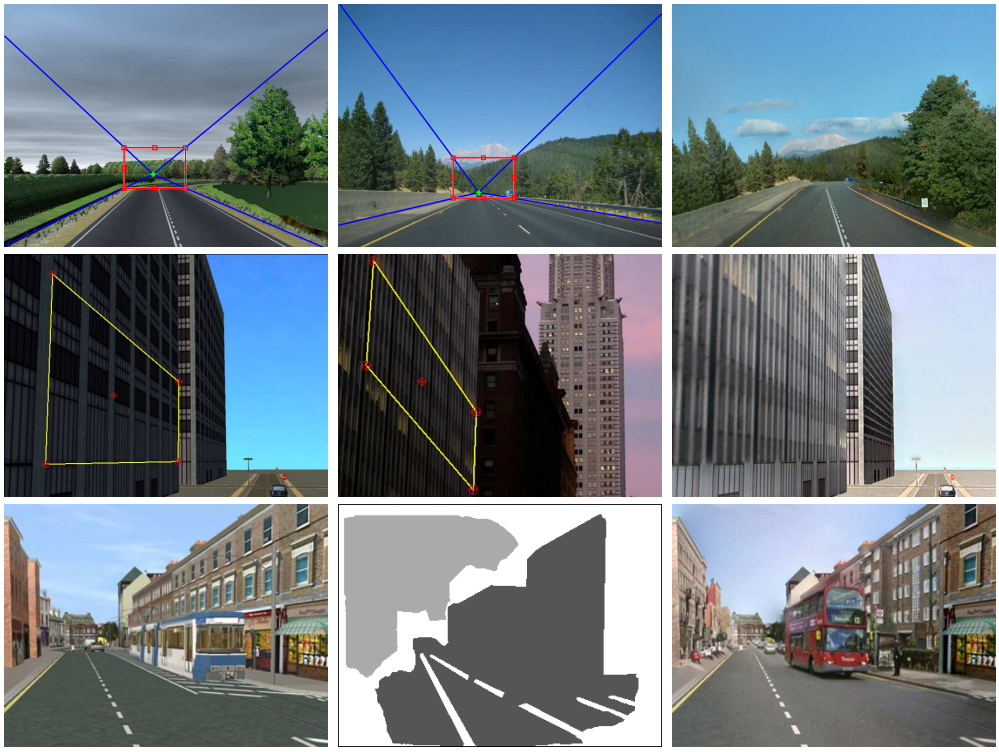

Figure 10: *CG2Real results on structured scenes. Top and middle rows: the real images (middle) were warped to align them with the corresponding CG images (left) based on coarse scene geometry specified by the spidery mesh (overlaid, top row) and rectangle (overlaid, middle row) to generate the results on the right. Bottom row: light and dark gray scribbles (middle) indicate that particular real image matches should not appear in those regions, and white areas indicate that these regions in the CG image (left) should be protected in the final result (right).*

LALONDE, J.-F., HOIEM, D., EFROS, A. A., ROTHER, C., WINN, J., AND CRIMINISI, A. 2007. Photo clip art. *ACM Transactions on Graphics 26*, 3 (July), 3:1–3:10.

LAZEBNIK, S., SCHMID, C., AND PONCE, J. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*.

LI, Y., SHARAN, L., AND ADELSON, E. H. 2005. Compressing and companding high dynamic range images with subband architectures. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH) 24*, 3, 836–844.

LIU, X., WAN, L., QU, Y., WONG, T.-T., LIN, S., LEUNG, C.-S., AND HENG, P.-A. 2008. Intrinsic colorization. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia) 27*. 152:1-152:9.

LOWE, D. G. 1999. Object recognition from local scale-invariant features. In *ICCV*.

LYU, S., AND FARID, H. 2005. How realistic is photorealistic? *IEEE Trans. Signal Processing 53*, 2, 845–850.

OLIVA, A., AND TORRALBA, A. 2001. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. Journal of Computer Vision 42*, 3, 145–175.

PARIS, S., AND DURAND, F. 2007. A topological approach to hierarchical segmentation using mean shift. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1–8.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics 22*, 3 (July), 313–318.

PITIÉ, F., KOKARAM, A. C., AND DAHYOT, R. 2005. N-dimensional probablility density function transfer and its application to colour transfer. In *Proc. IEEE Int. Conf. Computer Vision*, 1434–1439.

REINHARD, E., ASHIKHMIN, M., GOOCH, B., AND SHIRLEY, P. S. 2001. Color transfer between images. *IEEE Computer Graphics & Applications 21*, 5 (Sept./Oct.), 34–41.

ROSALES, R., ACHAN, K., AND FREY, B. 2003. Unsupervised image translation. In *Proc. IEEE Int. Conf. Computer Vision*.

ROTH, S., AND BLACK, M. J. 2005. Fields of experts: A framework for learning image priors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 860–867.

ROTHER, C., MINKA, T., BLAKE, A., AND KOLMOGOROV, V. 2006. Cosegmentation of image pairs by histogram matching—incorporating a global constraint into MRFs. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*.

RUSSELL, B. C., TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2008. Labelme: a database and web-based tool for image annotation. *Int. Journal of Computer Vision 77*, 1–3 (May), 157–173.

SIVIC, J., KANEVA, B., TORRALBA, A., AVIDAN, S., AND FREEMAN, W. T. 2008. Creating and exploring a large photorealistic virtual space. In *CVPR workshop on Internet Vision, 2008, archived in IEEE digital library*.

TAI, Y.-W., JIA, J., AND TANG, C.-K. 2006. Local color transfer via probabilistic segmentation by expectation-maximization. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 747–754.

WEISS, Y., AND FREEMAN, W. T. 2007. What makes a good model of natural images? In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*.

WEN, C.-L., HSIEH, C.-H., CHEN, B.-Y., AND OUHYOUNG, M. 2008. Example-based multiple local color transfer by strokes. *Computer Graphics Forum (Proc. of Pacific Graphics) 27*, 7, 1765–1772.

ZHANG, J., MARSZALEK, M., LAZEBNIK, S., AND SCHMID, C. 2007. Local features and kernels for classification of texture and object categories: a comprehensive study. *Int. Journal of Computer Vision 73*, 2, 213–238.